

SERIES 6000
DISC OPERATING SYSTEM (DOS-II)
GENERAL SPECIFICATION

Reissued
July, 1972

Datacraft Corporation

1200 N. W. 70th Street P. O. Box 23550 Fort Lauderdale, Florida 33307 (305) 974-1700

LIST OF EFFECTIVE PAGES

TOTAL NUMBER OF PAGES IN THIS PUBLICATION IS 97
CONSISTING OF THE FOLLOWING:

Page No.	Change No.	Page No.	Change No.	Page No.	Change No.
Title	Original				
A	Original				
i thru v	Original				
1-1 thru 1-3	Original				
2-1 thru 2-8	Original				
3-1 thru 3-32	Original				
4-1	Original				
5-1 thru 5-23	Original				
6-1 thru 6-5	Original				
7-1 thru 7-12	Original				
A-1 thru A-2	Original				
B-1	Original				
C-1 thru C-3	Original				

Original Issue,
July, 1971

Upon receipt of the second and subsequent changes to this technical document, personnel responsible for maintaining this publication in current status will ascertain that all previous changes have been received and incorporated. Action should be taken promptly if the publication is incomplete.

Datacraft Corporation

CONTENTS

<u>Section</u>	<u>Page</u>
I GENERAL DESCRIPTION	
1-1 Scope	1-1
1-2 Characteristics	1-1
II RESIDENT SYSTEM SERVICES	
2-1 General	2-1
2-2 Input/Output	2-1
2-3 ABORT	2-2
2-4 EXIT	2-2
2-5 HOLD	2-3
2-6 CHAIN	2-3
2-7 INFO	2-3
2-8 O/M (Operator Message)	2-5
2-9 Executive Traps	2-5
2-9.1 Power Fail/Restore Option	2-5
2-9.2 Program Restrict/Instruction Trap	2-5
2-9.3 Stall Alarm	2-6
2-9.4 Interval Timer	2-6
2-9.5 SAU Overflow/Underflow Trap	2-6
2-10 Foreground Debug	2-6
III NON-RESIDENT SYSTEM SERVICES	
3-1 General	3-1
3-2 Job Control	3-1
3-3 File Editing	3-4
3-3.1 Processor File Editor	3-4
3-3.2 Name File Editor	3-5
3-3.3 Source File Editor	3-6
3-3.4 Library File Editor	3-7
3-3.5 Library File Editor Error Message Codes	3-10
3-4 Object Time Trace	3-11
3-5 Debug	3-13
3-6 Link Loader	3-14
3-6.1 \$LINK Statement	3-14
3-6.2 Linking Across the 32K Boundary	3-16
3-6.3 Common Memory Allocation	3-17
3-6.4 Block Data Subprograms	3-17
3-6.5 Options	3-17
3-6.6 Link Map	3-18
3-6.7 Code Processing	3-19
3-6.8 Error Messages	3-23
3-6.9 Link Examples	3-25
3-6.10 Input and Code Placement	3-29

CONTENTS (Cont'd.)

<u>Section</u>		<u>Page</u>
IV	BACKGROUND PROCESSORS AND UTILITIES	4-1
V	LOGICAL FILES AND PHYSICAL DEVICES	
5-1	File/Device Assignments	5-1
5-1.1	File Assignments	5-1
5-1.2	Device Assignments	5-2
5-1.3	Blocked File Handling.	5-3
5-2	Input/Output Functions	5-8
5-2.1	All Devices	5-8
5-2.2	Disc	5-10
5-2.3	Magnetic Tape	5-12
5-2.4	Console Keyboard	5-14
5-2.5	Console Printer	5-15
5-2.6	Paper Tape Reader	5-16
5-2.7	Paper Tape Punch	5-18
5-2.8	Card Reader	5-19
5-2.9	Card Punch.	5-21
5-2.10	Line Printer.	5-22
VI	OPERATING PROCEDURES	
6-1	Loading Procedures	6-1
6-2	Operator Control Facilities	6-1
6-2.1	Abort Procedure.	6-1
6-2.2	Hold Recovery Procedure	6-2
6-3	Using System Processors.	6-2
6-4	Operator Messages.	6-2
VII	SYSTEM GENERATION	
7-1	General.	7-1
7-2	Preliminary Procedures	7-1
7-2.1	External Definition Table.	7-1
7-2.2	External Equivalence Table	7-1
7-2.3	Service Linkage Table	7-2
7-2.4	Miscellaneous Parameter Table	7-2
7-2.5	Priority Interrupt Linkage Table.	7-4
7-2.6	I/O Handler Linkage Table	7-4
7-2.7	Physical Assignment Table	7-6
7-2.8	System Service Directory	7-6
7-2.9	Disc Extents Table	7-6
7-2.10	Tape Options Table.	7-6
7-3	System Generation Procedures.	7-6
7-3.1	SGS Loading Procedure	7-6
7-3.2	SGS Operating Procedure	7-7

CONTENTS (Cont'd.)

<u>Section</u>		<u>Page</u>
VII	SYSTEM GENERATION (Continued)	
7-3.3	Processor File Development	7-7
7-3.4	Library File Development.	7-10
7-3.5	Name and Source File Initialization.	7-11
7-3.6	Processor File Expansion	7-11
7-3.7	File Listing	7-11
APPENDIX A	Physical Characteristics of the Moving-Head Disc	A-1
APPENDIX B	Channel, Unit, and Interrupt Assignments	B-1
APPENDIX C	Sample System Generation Job Stream	C-1

ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
3-1	Code Placement Format	3-30
5-1	Block Structure	5-7
5-2	Memory Map	5-8
7-1	Paper Tape Bootstrap	7-7
A-1	Cylinder and Track Relationships	A-2

TABLES

<u>Table</u>	<u>Page</u>
1-1 DOS Memory Map	1-2
2-1 Resident Services	2-1
2-2 Executive Traps	2-5
2-3 SAU Trap Control	2-7
3-1 Job Control Statements	3-2
3-2 Library File Editor Error Messages	3-10
3-3 Debug Control Statements	3-15
3-4 Values of Link Parameters	3-16
3-5 Link Loader Error Codes	3-23
3-6 Link Loader Input Codes	3-31
3-7 Link Loader Special Action Codes	3-32
5-1 Standard Logical File Numbers	5-1
5-2 Standard Physical Device Table	5-3
5-3 Valid I/O Function Codes	5-9
5-4 Line Printer Carriage Control Characters	5-22
6-1 DOS Operator Messages	6-3
7-1 Standard Assignments, Service Linkage Table	7-2
7-2 Miscellaneous Parameter Table	7-3
7-3 Standard Assignments, Priority Interrupt Linkage	7-4
7-4 Standard Assignments, I/O Handler Linkage Table	7-5
A-1 System Characteristics	A-1
B-1 Standard Channel/Unit and Interrupt Assignments	B-1

SECTION I

GENERAL DESCRIPTION

1-1 SCOPE

This document contains the basic operational and programming considerations pertinent to the Series 6000 Disc Operating System (DOS-II).

1-2 CHARACTERISTICS

DOS is a real time, hardware-interrupt oriented, operating system designed for flexibility and efficiency. Its organization, modularity and interrupt orientation permit custom system development. DOS consists of a System Linkage Module, Foreground Executive Module, System Service Module, I/O Control System, Link Loader, Debug, and Disc Edit Modules. Object Time Trace is a service that is optional with the system.

The I/O Control System, which remains resident with the other pertinent services, is logical-file oriented and employs a simple scheme for communicating with background or foreground. Single or double buffering is allowed under this scheme. The I/O Control System detects hardware trouble and user errors and initiates appropriate messages to the console teletypewriter. Job Control, Debug, Trace, Link Loader, and Disc Edit Modules are loaded from disc into a common non-resident service area of memory, when required. All processors and utilities are I/O independent and function in the background under DOS. (A memory map of DOS is shown in Table 1-1.)

Disc storage is divided into logical files (11 through n). Files 11 through 14 are restricted system files and 15 through n are background work files. The processors on file 11 are in load module format. These processors may be requested externally through Job Control or internally through the system service CHAIN (refer to Paragraph 2-6). A directory gives direct access to load modules on the file.

File 12, Library File, contains subprograms only in link module format. This file is searched by the Link Loader to satisfy external subroutine requests during a linking process. File 13, Name File, is reserved for user programs or subroutines in link module format. File 14, Source File, is for symbolic modules (card image) in a compressed and blocked format. File 15, Link-Ready File, is a temporary work file for preparing link modules for linking or adding to file 12 or 13. This file should be assigned to binary output for compilation or assembly. Files 16 through 22, Work Files, are for general use by background processors.

DOS uses the absolute sector number in referencing a specific location on the disc (refer to Appendix A). The sector number is converted to a disc address before executing the command word. The next available record address is calculated from the requested word count and the current record address. This new address is tested against the extents of the file before the disc I/O function is executed for the current address.

Table 1-1. DOS Memory Map

Memory High	
Background High	<p>Foreground Work Area</p> <p>This is a variable size area of memory (0 to n locations) which may be defined and accessed by foreground as desired. This area is also used by the memory file handler if included in the system.</p>
Background Low	<p>Background Area</p> <p>This is a restricted area of memory allocated to background processors. If the program restrict option is active, a processor in this area may not alter memory outside its bounds. System services are accessible from background through the Service Linkage table.</p>
Approximately 330010 Locations	<p>Non-Resident System Service Area (approx.1150 locations)</p> <p>This area is reserved for Job Control, Link Loader, Debug, Trace, File Edit and such system services which work in an overlay fashion.</p>
	<p>Resident System Service Module (approx.300 locations)</p> <p>This module includes system information and initialization services, chain loader, etc.</p>
	<p>Resident I/O Control System (approx.1500 locations)</p> <p>The I/O control system contains an I/O control routine plus a handler for each peripheral device specified in the handler linkage table.</p>
	<p>Foreground Executive Modules (0 to - n locations)</p> <p>Foreground executive modules are user defined routines for desired foreground processing plus executive interrupt processors.</p>
	<p>Resident System Linkage Module (approx.200 locations)</p> <p>This module includes Service Linkage table, I/O handler linkage table, Physical Assignment table, Priority Interrupt Linkage table, disc file extents and miscellaneous parameters.</p>
Memory Low	

A table of file extents is maintained in the resident system. This table contains the first and last sector number for each disc file. The user may, at system generation, allocate disc storage to suit the requirements of his particular system. Disc allocation may also be modified via the \$EXTENTS statement of Job Control (refer to Table 3-1 of Section III of this document).

Bookkeeping for sequential file processing is performed by the disc handler, with a full complement of logical functions allowed (refer to Paragraph 2-2). Special functions allow random access of a file once it has been sequentially defined.

SECTION II

RESIDENT SYSTEM SERVICES

2-1 GENERAL

The resident system services are those system functions required to serve background programs. Table 2-1 is a list of the resident system services and their functions. Background programs may use these services through a Branch and Link Unrestricted (BLU) instruction. The referenced service is accessed through the Service Linkage table contained in memory locations 0 through 37 (refer to Paragraph 7-2.3).

Table 2-1. Resident Services

Linkage Address (Octal)	Identification	Function
0	BLU \$ABORT	Abort current background job.
1	BLU \$I/O	I/O control routine.
2	BLU \$EXIT	Return to Job Control.
3	BLU \$HOLD	Output operator message and wait.
4	BLU \$CHAIN	Load absolute program module.
5	BLU \$INFO	Return system information.
6	BLU \$O/M	Stack an operator message and continue.

2-2 INPUT/OUTPUT

The Resident I/O Control System performs all I/O functions on a priority interrupt basis, providing the ability to overlap I/O operations and internal processing. Normal I/O requests are made on logical files; physical devices may be assigned to logical files through Job Control. Normal I/O requests are made via the following sequence of instructions:

ALPHA	TLO BLU BON	PARLIST \$I/O ALPHA	<ul style="list-style-type: none"> . (K) = address of parameter list. . call I/O control. . repeat request if busy. . transfer has been initiated and will be performed on an interrupt basis concurrent with processing.
PARLIST	...	'XXYY n	<ul style="list-style-type: none"> . XX=logical file, YY=function code. . word count.
BUFFER	DAC BLOK	BUFFER n	<ul style="list-style-type: none"> . buffer address. . reserve n words of storage.

DOS-II General Specification

If a handler is busy when a request is received, a negative condition is returned to ALPHA+2. If not busy, the request is interrogated for validity. On detecting an error, message "E@XXXXXX" is typed and processing is held in an idle state. The address XXXXXX points to the invalid call (ALPHA+1), allowing the operator to make a correction via Debug and continue. On being released by the operator, the handler returns a negative condition to ALPHA+2 forcing the request to be repeated using the corrected parameters. If it is not desirable to continue, the operator may abort the job.

Valid requests are initiated and a zero condition returned to ALPHA+2. This allows processing to continue concurrent with a data transfer. Functions not requiring a parameter list (word count and buffer address) may be executed via the following sequence:

ALPHA	TNK	'XYY	. (K)=(logical file/function code).
	BLU	\$I/O	. Call I/O control.
	BON	ALPHA	

This sequence may be employed to test status, open a file, etc.

Since operations are made on a priority interrupt basis and each handler maintains its own busy/not busy status, multiple I/O functions on different devices may be in process concurrently. Acceptable logical files (XX) and function codes (YY) are described in Section V of this document.

2-3 ABORT

The ABORT function provides the means whereby the current job can unconditionally be terminated, either by program control or operator intervention (refer to Paragraph 6-2.1 for operator initiated Abort considerations). When an Abort command is executed, all I/O is terminated, the status of all I/O handlers is reset, and all priority interrupts are disabled except those relating to the Operator Communications Device (OCD). The message "ABORT" is printed on the OCD and control is transferred to Job Control. ABORT may be initiated under program control by the following calling sequence:

BLU	\$ABORT
-----	---------

When ABORT is program initiated, the Job Stream logical device is not re-assigned.

2-4 EXIT

The EXIT function provides the means to terminate a background process without affecting any I/O operation that may be in progress. EXIT may be initiated only by program control. When EXIT is program initiated, the Job Stream logical device is not re-assigned.

Assembly Language Call

BLU	\$EXIT
-----	--------

FORTTRAN Call

CALL EXIT

2-5 HOLD

HOLD is an operator communications service wherein a process may be placed in an "idle" state while a specified message is output to the OCD. The following is the calling sequence to initiate the HOLD function:

	TMK	MESSAGE
	BLU	\$HOLD
MESSAGE	i XX	= "message text"

where: XX is the octal word count of the text message.

By depressing the control key BELL, the operator may release the current "idle" state and return control to the point from which the last hold was issued (refer to Paragraph 6-2.2).

2-6 CHAIN

CHAIN allows a large program to be segmented into any number of absolute load modules and "chain loaded" under program control for execution. Different Chain modules may reference a common data pool and may call each other in any order. The last executable module should call EXIT. The CHAIN calling sequence is:

<u>Assembly Language</u>	<u>FORTRAN</u>
BLU \$CHAIN	CALL CHAIN (6HXXXXXX)
DAC = "XXXXXX"	

where: XXXXXX is the identification of the module being called.

To initiate a CHAIN sequence, the Job Control statement "LOADGO XXXXXX" is issued (XXXXXX is the name of the first load module to be executed).

2-7 INFO

INFO provides background processors with system information such as date, lines per page, options, flags, etc. An information request is as follows:

TOK	n
BLU	\$INFO

DOS-II General Specification

<u>Value of n</u>	<u>Result</u>	<u>References</u>
1	The 24-bit Option word is returned in the A register.	Table 3-1 \$OPTIONS
2	The 9-character date is returned in the E, A, and I registers.	Table 3-1 \$DATE
3	The lines per page integer is returned in the A register.	Table 3-1 \$LINES
4	The 24-bit Flag word is returned in the A register.	Table 3-1 \$FLAGS
5	The Current 6-character job name is returned in the D register.	Table 3-1 \$JOB
6	The double precision integer for the current run time is returned in the D register. The time is in microseconds.	
7	The background-low address is returned in the E register and background-high address is returned in the A register. The non-resident service address (NRS) is in I and memory high is in J. The operator communications linkage address, F. CP, is returned in register K. (Refer to Table 7-2).	Table 3-1 \$BG. HI
8	The link parameter pointers are returned in registers I, J, and K. I contains the first address of the following consecutive parameters: p. name, e. low, e. high, e. start, l. low, l. high, and c. base. J contains the first address of the System Service Directory and K contains the last System Service Directory Address, plus one. This information is for system processors such as the Link Loader.	Table 3-1 \$LINK
9	The disc file extents associated with the logical file number specified in register I are returned in registers E and A.	Table 3-1 \$EXTENTS

The physical device number assigned to logical file I is returned in I. The condition register is negative for an invalid device number and zero otherwise. E and A are zero if the specified file is not assigned to a disc file. Register J is negative if the specified file is restricted.

2-8 O/M (Operator Message)

O/M is an operator communications service whereby a process may stack an operator message using the following abbreviated calling sequence.

```

      TMK      MESSAGE
      BLU      $O/M
MESSAGE      'XX'      ="message text"

```

where: XX is the octal word count of the text message.

2-9 EXECUTIVE TRAPS

The executive trap routines are a part of the Foreground Executive Module. The standard interrupt assignments for the executive traps are defined in Table 2-2.

Table 2-2. Executive Traps

Linkage Address (Octal)	Group, Level	Function
60	0,0	Power Down
61	0,1	Power Up
62	0,2	Program Restrict
63	0,3	Instruction Trap
64	0,4	Stall Alarm
65	0,5	Interval Timer
66	0,6	SAU Overflow/Underflow
67	0,7	Address Trap

2-9.1 Power Fail/Restore Option

When power fails, the Power Fail/Restore routine saves all registers and halts the CPU. When power returns, a "Power Failure, Release to Continue" message is output to the Operator Communications Device. A release will restore registers and continue the operation of the current program.

2-9.2 Program Restrict/Instruction Trap

When the Program Restrict key switch is enabled, the operating system has two modes operation. The non-resident services such as Job Control, Link Loader, File Edit, and Trace operate in an Unrestricted mode; that is, these services have access to any location in memory. All other non-resident services or background programs operate in the Restrict/Unprivileged mode;

DOS-II General Specification

that is, the programs may not reference any locations below the "execution low" address of the program and above "background high". Programs operating in the Restrict/Unprivileged mode are prevented from executing certain instructions as defined in the Computer System Reference Manual.

When a Program Restrict violation occurs the HOLD message "PR VIOLATION ((XXXXXX" is output to the Operator Communications Device where "XXXXXX" is the address where the violation occurred. A release will generate an Abort.

When an Instruction Trap violation occurs, the HOLD message "IT VIOLATION ((XXXXXX" is output where "XXXXXX" is the address where the violation occurred. A release will generate an Abort.

2-9.3 Stall Alarm

The Stall Alarm is enabled or disabled by the Stall Alarm key switch.

When a Stall Alarm violation occurs the HOLD message "SA VIOLATION ((XXXXXX" is output to the Operator Communications Device where "XXXXXX" is the address where the violation occurred. A release will generate an Abort.

2-9.4 Interval Timer

This option provides the system with a run-time service. This time is maintained as a double-integer in microseconds; zero when the system is initialized and updated, using the T-register, each second.

A processor may request the current time through the system service INFO as follows:

TOK	6
BLU	\$INFO

which returns time in register D, accurate to within 10 microseconds relative to the BLU.

This INFO service is used by the \$JOB and \$EOJ control statements to compute Job-time.

2-9.5 SAU Overflow/Underflow Trap

2-3. The SAU Trap routine processes overflow/underflow conditions as described in Table

2-10 FOREGROUND DEBUG

Foreground Debug is a resident system service that can be used to output memory locations in either octal or ASCII text. A register or any location in memory can be modified with an input statement. Breakpoints can be set and reset in any area of memory. Specific data configurations can be scanned for in memory. Refer to Section III for command definitions.

Table 2-3. SAU Trap Control

Option	Function
15=1 15=0	DISABLE TRAP ENABLE TRAP
14=1 14=0 13=1 13=0	SUPPRESS UNDERFLOW HOLD MESSAGE OUTPUT UNDERFLOW HOLD MESSAGE "SAU 4" NO ABORT ON UNDERFLOW; return X=F. P. Z ABORT ON UNDERFLOW
12=1 12=0 11=1 11=0	SUPPRESS OVERFLOW HOLD MESSAGE OUTPUT OVERFLOW HOLD MESSAGE NO ABORT ON OVERFLOW ABORT ON OVERFLOW
	OVERFLOW MESSAGES AND RESULTS "SAU 1" if SQUARE ROOT (-X); return X=F. S. P "SAU 2" if FIX (X .GE. 2**23); return X=F. S. P "SAU 3" if DIVISION BY 0; return X=F. S. P "SAU 5" if EXPONENT OVERFLOW WITH POSITIVE MANTISSA; return X=F. S. P "SAU 6" if EXPONENT OVERFLOW WITH NEGATIVE MANTISSA; return X=F. S. N * "FER 6" if SIN or COS (X .GT. RANGE); return X=F. P. Z * "FER 1" if LOG (X .LE. 0); return X=F. P. Z * "FER 2" if EXP (X .GT. 2**7); return X=F. S. P * "FER 13" if ATAN2 (0, 0); return X=F. P. Z
10=0 *	OUTPUT MESSAGE "ABORT FER 12" and ABORT on FORMATTED READ ERROR
10=1 *	OUTPUT MESSAGE "HOLD FER 12" and HOLD on FORMATTED READ ERROR; retry after release
*	OUTPUT MESSAGE "FER 21" and ABORT if MORE THAN 10 FILES DEFINED FOR FORTRAN RANDOM I/O
*	OUTPUT MESSAGE "FER 22" and ABORT if ACCESS IS REQUESTED OF AN UNDEFINED RANDOM FILE
*	OUTPUT MESSAGE "FER 23" and ABORT if ACCESS IS REQUESTED OF A RECORD NUMBER WHICH IS NOT ON THE SPECIFIED RANDOM FILE

* Denotes a Software Triggered Interrupt.

F. S. P denotes Full Scale Positive value ('37777777, '37777577)

F. S. Z denotes Floating Point Zero value ('00000000, '00000201)

F. S. N denotes Full Scale Negative value ('60000000, '00000177)

DOS-II General Specification

Communication between the user and Foreground Debug must be activated with each statement through the use of the Operator Communications input key (rub out). Once activated a control statement can be input through the Operator Communications Device. The statement is processed and control returns to the interrupted process.

The Foreground Debug routine is a part of the Foreground Executive Module. Linkage of Foreground Debug with the operating system is optional at system generation (See Section 7).

SECTION III

NON-RESIDENT SYSTEM SERVICES

3-1 GENERAL

The non-resident system services are those service routines that reside on the Processor File. They are loaded via Job Control statements and overlay the Job Control area of the system. The Processor File has a directory that permits direct access to these routines. The directory is entered into the non-resident service area to obtain the information necessary to load and execute the routine.

The size of the non-resident service area is defined in the System Service module. Background area begins where the non-resident service area ends. This area must be large enough to contain any of the following routines: Job Control, Link Loader, EDITPF, and Debug. The size of the directory must not exceed this area. The following routines use this area, but may extend into the background area: INCLUDE, EDITLF, EDITNF, EDITSF, and Trace.

3-2 JOB CONTROL

Job Control is a non-resident routine without priority. It accepts and processes commands that monitor the "batch flow" of the system. Job Stream commands are identified by the presence of a dollar sign (\$) in character position one of a symbolic source statement. This implies that a background processor may not request input from the assigned Job Control device that has this identification configuration. Should such a request be inadvertently made, the assigned Job Control device handler moves the statement to a resident Job Control buffer and transfers control to Job Control. This function prevents one job from interfering with another job in the batch stream.

If a Job Control command or any of its associated parameters are invalid, the statement is rejected and the message "ICS" (Illegal Control Statement) is output to the OCD and a "hold" is performed. On being released (Control Key BELL), control is returned to Job Control for another statement.

The format of all Job Control statements is as follows: Column one (i.e., the first print position) must contain a dollar sign (\$); the next *j* columns contain the command followed by one space; and the next *k* columns contain the parameter(s) associated with the command. If there is more than one parameter to enter, commas will be used as separators. Imbedded blanks are not permitted in the parameter string. Table 3-1 contains a list of acceptable statements and a definition of each.

Table 3-1. Job Control Statements

Statement	Definition
\$JOB XXXXXX	The \$JOB statement opens a new job and saves the 6-character name (XXXXXX). The Abort flag is reset and the Restrict flag is set. (Once an Abort occurs, all subsequent Job Control statements are rejected until a Job statement is received.) The current job name may be acquired from background through the service info (refer to Paragraph 2-7).
\$ASSIGN L,P,L,P	This statement causes physical device 'P' to be assigned to logical file 'L'. To suppress operations on a logical file, the physical device number 0 is assigned to the particular file. For example, to suppress binary output, the statement would be: \$ASSIGN 5,0. Section V contains a description of logical file and physical device assignments for DOS.
\$DATE XXXXXXXXX	The 9-character text (XX:XXXXXX) is saved for subsequent reference by processing programs. The data is acquired by a background processor through the service info (refer to Paragraph 2-7).
\$LINES n	This command sets the lines per page count to the specified number (n is a decimal integer), overriding the lines per page count established when a \$JOB statement is encountered. The count is acquired by the service info (refer to Paragraph 2-7).
\$OPTIONS .0,1,2,3,...,n	This statement sets the specified bits of a memory word; i.e., places a 1 in the selected bit positions (n is a decimal integer less than or equal to 23). The leading dot (.), if present, enters zeros in all bits of the word prior to setting the specified bits. This word is acquired by the service info (refer to Paragraph 2-7).
\$FLAGS .0,1,2,3,...,n	This command functions in the same manner as the \$OPTIONS statement, except that a different memory location is used.
\$BG. HI n	The octal integer (n) is stored as "background high" for system use. Processor may acquire this parameter through the service info (refer to Paragraph 2-7). The area between "background high" and "memory high" is reserved for the memory file handler or for foreground use. By varying "background high", the user may expand or compress this area.
\$INCLUDE name	If the name parameter is absent, the Include statement accepts link modules from the binary input file (04) and places them on the link ready disc file (15). If a name is specified, the include processor searches the name file for the specified name, then moves the link module to the link ready file. When the Include processor encounters an END\$ record, an end-of-file is written and a record backspaced on the link ready file in anticipation of subsequent Include statements. If a name is specified, only one link module is moved from the name file. Should additional link modules be included within the same job, then the last end-of-file will be overlaid with the newly encountered link modules. When any Link statement is encountered, the link ready file is rewound and all link modules are loaded until an end-of-file is encountered. Upon encountering the end-of-file, the link loader scans the library file for any unsatisfied external requests.
\$LINK l.low,l.high,e.low,c.base	If the parameters are specified, they are stored for subsequent reference by the Link Loader. Parameters "l.low" and "l.high" define an area of memory in which the core image of an executable program is built from all link modules residing on the link ready file (15), plus any library routines they require from the library file (12). The upper portion of this link area is used by the Link Loader as temporary storage for developing and external linkage table. The parameter "e.low" is a relocation bias for the program being loaded into the link area. If "e.low" is not specified, it is assumed to be equal to "l.low". By specifying different addresses for "e.low" and "l.low", a program may be built in one area of memory to be executed in another area. For example, a program may be linked at 10000, relative to 20 and dumped (see \$DUMP) in bootstrap format. If "l.low" and "l.high" are outside the bounds of Background and the system is restricted (see \$RESTRICT), the Hold message (ICS) is typed and the statement is ignored. If "l.low" and "l.high" are not specified, "background low" and "background high" are assumed and "e.low" is assumed to be equal to "background low". On completion of a link load, the parameters execution low, execution high, program start, and program name (if any) are retained by the system for subsequent use by the \$GO, \$EDITPF, \$DUMP, or \$DUMPF command processors. (Refer to Paragraph 3-6 for a description of c.base.)
\$LINKOL	The \$LINKOL (Link Overlay) statement performs the \$LINK function in the area specified by background parameters with p.low set to overlay the non-resident service area after the link process is completed. A \$GO is automatic.
\$GO	The program restricts registers are set to "background low" and "background high" and control is unconditionally transferred to the starting address (p.start) of the current-resident background program.
\$LINKGO l.low,l.high,e.low,c.base	This statement combines the action of \$LINK and \$GO.
\$RESTRICT	This statement sets the Restrict flag within the resident system. When the flag is set, writing is not permitted on system disc files.
\$ALLOW	This statement resets the Restrict flag. When the flag is reset, the system disc files can be modified.
\$LOADGO name	This command causes the processor file directory to be scanned for the specified name. If the name is found, the program is loaded from the processor file and executed. If the name is not found on the processor file directory, an error message, "ILR" is typed and a Hold condition is entered. The operator may correct the card and release the Hold, or abort the program. Note that this command applies only to programs on the processor file.
\$HOLD text	When this command is executed, the text is typed and the system enters a Hold condition until the operator intervenes.
\$EOJ	This statement signifies the end of the current job. When executed, this statement causes the current job name and its running time in hours, minutes, and seconds to be output to the assigned list out file.

Table 3-1. Job Control Statements (Cont'd.)

Statement	Definition										
\$DUMP p.name,e.low e.high,p.start, d.low	<p>An absolute load module is generated on the assigned binary output file. A load module consists of two binary records. Record 1 contains the following six words:</p> <table> <tr> <td>Words 1 and 2</td><td>6-character name</td></tr> <tr> <td>Word 3</td><td>execution low</td></tr> <tr> <td>Word 4</td><td>execution high</td></tr> <tr> <td>Word 5</td><td>program start</td></tr> <tr> <td>Word 6</td><td>checksum of program low through programs high</td></tr> </table> <p>Record 2 consists of n words (p.high-p.low + 1) from memory, starting at "d.low". If "d.low" is not specified, it is assumed to be equal to "p.low". If parameters are not specified, the parameters established by the last \$LINK are used. A name may only be specified and all other parameters will be assumed.</p>	Words 1 and 2	6-character name	Word 3	execution low	Word 4	execution high	Word 5	program start	Word 6	checksum of program low through programs high
Words 1 and 2	6-character name										
Word 3	execution low										
Word 4	execution high										
Word 5	program start										
Word 6	checksum of program low through programs high										
\$DUMPBF p.name,e.low, e.high,p.start, d.low	This function is the same as \$DUMP except that record 1 (parameter record) is suppressed, producing a bootstrap format.										
	<p>NOTE</p> <p>The following services permit the use of IOCS via job Control statements (refer to Paragraph 5-2 for descriptions of the functions performed).</p>										
\$OPEN xx	Open logical file xx (octal).										
\$CLOSE xx	Close logical file xx (octal).										
\$REW xx	Rewind logical file xx (octal).										
\$BSF xx	Backspace a file on logical file xx (octal).										
\$ADF xx	Advance a file on logical file xx (octal).										
\$RPF xx	Reposition current file on logical file xx (octal).										
\$WEF xx	Write End-of-File on logical file xx (octal).										
\$XXYY xxyy	Perform function yy on logical file xx (octal).										
	<p>NOTE</p> <p>The following services are used for DOS system file maintenance (refer to Paragraph 3-3 for descriptions and use of these functions).</p>										
\$EDITPF	Call processor file edit routine in the non-resident service area.										
\$EDITLF	Call library file edit routine into the non-resident service area.										
\$EDITNF	Call name file edit routine into the non-resident service area.										
\$EDITSF	Call source file edit routine into the non-resident service area.										
\$EXTENTS xx, S1, S2, R	<p>Establishes the file extents for logical disc file xx.</p> <p>S1 = first record address (decimal sector #). S2 = last record address. R = restrict code.</p> <p>The presence of the letter "R" restricts the file.</p>										
\$DEBUG	This statement causes the Debug program to be loaded in the non-resident area of memory (refer to Paragraph 3-5 for descriptions of the Debug commands).										
\$TAPEOP transport, mode, density, cpw	<p>Transport is 0, 1, 2, 3. mode is: 0 for Binary or ASCII - symbolic. 1 for BCD or EXTENDED BCD. density is: 0 for PEC-LOW or CDC-200 cpi. 1 for PEC-HI or CDC-556 cpi. cpw is: 1 for 1 cpw. 2 for 2 cpw. 3 for 3 cpw. 4 for 4 cpw.</p>										
\$REPSYS	This command replaces the DOS system on file 11 with the resident DOS.										

3-3 FILE EDITING

There are four non-resident disc file edit processors. These processors permit the user to edit the system Processor File (11), Library File (12), Name File (13), and Source File (14) through Job Control.

The file edit processors are loaded for execution via the Job Control statements \$EDITPF, \$EDITLF, \$EDITNF, and \$EDITSF. Once loaded, the processor requests commands from the Job Control logical device (00). Commands must start in column one and be terminated by a blank. If a name is used, it must appear as six consecutive characters following the command delimiter (a blank). The EXIT command causes the processor to return to Job Control.

3-3.1 Processor File Editor

The processor file (11) editor is loaded into the non-resident service area by the \$EDITPF Job Control statement.

Programs in link module format can be added to the processor file (11) with the following procedure:

- 1) Rewind the link ready file (15) and place a link module (a main program and any subroutines required) on file 15 via an assembly, compilation or the \$INCLUDE Job Control statement.
- 2) Link the program with a \$LINK Job Control statement as illustrated in paragraph 3-6.1. Any requested library routines are linked from the Library File.
- 3) Add the result of the link to the processor file (11) via the following statements:

```
$ EDITPF  
ADD name
```

The added load module may be loaded and executed by the \$LOADGO command.

The processor file (11) has a directory that allows direct access of records. The directory is read into the background area and updated when the Add, Delete, Replace, and Squeeze Commands are used. This means background is destroyed for all EDITPF functions except List.

EDITPF error messages are defined in Table 6-1. The processor file (11) is rewound by the editor for each command. The following commands can be used with EDITPF.

ADD name	If "name" is absent, the load module presently in memory is added to the processor file (11), using the name generated via the NAME pseudo-op. If "name" is present, it is used as the name of the module being added. An operator message will be typed if a processor with the specified name is already in the processor file (11), or if the load module is not named.
----------	--

DELETE name	The load module name specified by the name parameter is deleted from the processor file (11). The name parameter is required.
-------------	---

REPLACE name	Replace is a combination of Delete and Add. The presence or absence of the parameter causes the same action to be taken as with the Add command.
SQUEEZE	The processor file (11) is compressed, eliminating all previously deleted modules. Work file 16 is used as a temporary file for the Squeeze function.
LIST	The names of all modules on the processor file (11) are listed on the list output file (6). Deleted modules that still reside on the file can be determined by a blank name in the output list.

NOTE

Do not ABORT the execution of Add, Replace, or Squeeze commands because the physical end-of-file may be lost by this action.

3-3.2 Name File Editor

The name file (13) editor is loaded into the non-resident service area via the \$EDITNF Job Control statement.

The name file (13) is used to store main programs in link format. The INCLUDE processor is used to move a link module from the name file (13) to the link ready file (15). The name file editor and the INCLUDE processor can be used with blocked or unblocked files. (Refer to blocked file handling.)

A name file (13) module is comprised of link format records that must begin with a name definition in the first record and end with an END code in the last record. Name file modules are identified by their name definitions as explained below.

A name definition is generated by the Macro Assembler when the pseudo-operation "NAME" is encountered, or by the FORTRAN IV Compiler when a "NAME" statement is received. If the module does not have a name definition, the first external definition in the link module may be used to identify the module.

EDITNF error messages are defined in Table 6-1. The name file (13) is rewound by the editor for each command. The link ready file (15) is rewound by the editor when required. The following commands can be used with EDITNF.

ADD name	The link module specified by the name parameter which presently resides on the link ready file (15) is added to the name file (13) using the name which was generated via the NAME pseudo-op or an XDEF. The name parameter is required.
DELETE name	The link module specified by the name parameter is deleted from the name file (i.e., the NAME pseudo-op or the first XDEF of the module is changed to a blank name).
REPLACE name	Replace is a combination of a Delete and Add. Presence or absence of the parameter causes the same action to be taken as with the Add command.

DOS-II General Specification

SQUEEZE	The name file (13) is compressed, eliminating all previously deleted modules. Work file 16 is used as a temporary file for the Squeeze function.
LIST	The names of all modules on the name file (13) are listed on the list output file (6). Deleted modules that still reside on the file can be determined by a blank name in the output list.

NOTE

Do not ABORT the execution of Add, Replace, or Squeeze commands because the physical end-of-file may be lost by this action.

3-3.3 Source File Editor

The source file (14) editor is loaded into the non-resident service area by the \$EDITSF Job Control statement.

The source file (14) is for symbolic modules in a compressed and blocked format. EDITSF error messages are defined in Table 6-1. The source file (14) is rewound by the editor for each command. The work file (16) is rewound initially and when required.

The following commands can be used with EDITSF.

ADD name	The source module on logical file 16 is compressed and added to the source file (14) and given the six-character name specified by the name parameter. The name parameter must be present. The Add function adds from file 16 until an end-of-file is reached.
DELETE name	The source module specified by the name parameter is deleted from the Source File (i.e., the module name is changed to a blank).
REPLACE name	Replace is a combination of Delete and Add. The name parameter is required.
SQUEEZE	The source file (14) is compressed, eliminating all previously deleted modules. Work file 16 is used as a temporary file for the Squeeze function.
LIST	The names of all modules on the source file (14) are listed on the list output file (6). Deleted modules that still reside on the file can be determined by a blank name in the output list.
UNBLOCK name	The source module specified by the name parameter is searched for on the source file (14). When found, it is copied onto logical file 16 in unblocked format (27 word records). An end-of-file is written following the source module on file 16. File 16 is rewound initially by the EDITSF statement but not for additional unblock commands. Additional source modules may be unblocked to file 16 with end-of-files separating each module.

If the name parameter is absent, the entire source file is unblocked onto file 16.

COPY

The entire source file is copied in blocked format from file 14 of file 16.

NOTE

Do not ABORT the execution of Add, Replace, or Squeeze commands because the physical end-of-file may be lost by this action.

3-3.4 Library File Editor

The library file (12) editor is loaded into the non-resident service area via the \$EDITLF Job Control statement.

A library module is comprised of link format records that must begin with an external definition in the first record and end with an END code in the last record. Library modules are identified by their external definition names.

An external definition is generated by the Macro Assembler when the pseudo-operation "XDEF" is encountered and by the FORTRAN IV Compiler when the statements "SUBROUTINE" or "FUNCTION" are received.

Library modules on the link ready file (15) can be added to the library file (12). The library and link ready files are rewound by the editor for each command. File 16 should be assigned to a temporary work file before using the Order command.

The editor may be used with blocked or unblocked files. (Refer to blocked file handling).

The following commands can be used with EDITLF.

ADD name

Each module to be added must be externally defined at assembly or compilation time.

If "name" is absent, all modules on the link ready file (15) are added to the end of the library file (12). The Add function adds from the link ready file (15) until an end-of-file is reached.

If "name" is present, the link ready file is scanned and the module specified is added to the library file. Any external definitions within a module may be used as the referenced name.

DELETE name

The library module whose name is specified by the parameter is deleted from the library file (12). Any external definition within a module containing multiple definitions may be referenced to delete the module.

DOS-II General Specification

REPLACE name

Replace is a combination of Delete and Add. Each module to be added must be externally defined at assembly or compilation time. The modules on the library file (12) that have external definitions the same as those being added from the link ready file (15) are deleted.

If "name" is absent, all modules on the link ready file (15) are added to the end of the library file (12). The link ready file must be terminated by an end-of-file.

If "name" is present, the link ready file is scanned and the module specified is added to the end of the library file. Any external definition within a module can be used as its name.

RENAME name1, name 2

The library file (12) is scanned and the external definition specified as name1 is replaced by name2. Any external definition within a module can be used as name1.

ORDER

The modules on the library file (12) are ordered, that is, modules are arranged such that all external requests precede the module being requested. This allows the Link Loader to satisfy all external requests from the library file in one pass.

Deleted modules are eliminated from the library file. File 16 is rewound by the Order command and should be assigned to a temporary work file.

NOTE

Do not ABORT the execution of Add, Replace, Order, or Squeeze commands because the physical End-of-File may be lost by this action.

LIST

This command outputs to the list out file (06) a cross reference of the modules on the library file (12) and continues with an alphabetic sort of the external definitions with their corresponding module number. The format of the cross reference is as follows:

```
* XXX   JJJJJJ   KKKKKK   YYY       LLLLLL   UNDEF
      MMMMMMM
```

where:

- * indicates (if present) that the module is out of order (i.e., the module requests an external definition which was previously defined on the library file).

XXX is the sequence number of the module.

JJJJJJ is the first external definition in module XXX.

KKKKKK is the first external request in the module XXX.
 YYY is the module number where KKKKKK is defined.
 LLLLLL is the second external request in the module XXX.
 UNDEF if present indicates that the request LLLLLL is undefined on the library file.
 MMMMMM is the second external definition in module XXX.

The following example illustrates the list output format:

1	AAAAAA	BBBBBB	UNDEF		
2	DELETED				
3	CCCCCC				
* 4	EEEEEE DDDDDD	FFFFFF	5	CCCCCC	3
* 5	FFFFFF	AAAAAA	1		

AAAAAA	1
CCCCCC	3
DDDDDD	4
EEEEEE	4
FFFFFF	5

Note that module number two has been deleted causing module number one to have an undefined request. Note that module number four is out of order because request CCCCCC is defined in module number three.

A listing of the same library file after an Order command is given below:

1	EEEEEE DDDDDD	FFFFFF	3	CCCCCC	2
2	CCCCCC				
3	FFFFFF	AAAAAA	4		
4	AAAAAA	BBBBBB	UNDEF		

AAAAAA	4
CCCCCC	2
DDDDDD	1
EEEEEE	1
FFFFFF	3

3-3.5 Library File Editor Error Message Codes

During the editing process, extensive checking is performed to insure a valid library file. If an error condition is detected, a message will be output to the list output logical device (06) and the editing aborted.

The format of the error message is as follows:

ELF XX MOD YYY AAAAAA

where:

XX is a two-decimal digit error code. (Definitions of the error codes are given in the table below.)

YYY is a three-decimal digit module number in which the error occurred.

AAAAAA is the last encountered external definition. This will be blank if the definition cannot be determined.

Table 3-2. Library File Editor Error Messages

Number	Definition
1	There is insufficient background area to build the external definition and external request tables.
2	Invalid control statement.
3	An external definition is missing from the beginning of a module being added to the library file.
4	An attempt is being made to add a module which duplicates a name on the library file.
5	The specified name is not on the link ready file.
6	The word count was not complete when a binary input record was requested.
7	The modules being added will not fit on the library file. To increase the size of the library file refer to \$EXTENTS, Table 3-1.
8	A name was not specified in the control statement.
9	The specified name is not on the library file.
10	An attempt is being made to add modules which exceed the 1000 module limit.

Table 3-2. Library File Editor Error Messages (Continued)

Number	Definition
11	An invalid loader code was encountered on the input file.
12	A source program error was encountered on the input file.
13	An end-of-file was encountered at an improper position on the library file.
14	An end-of-file was encountered at an improper position on the link ready file.
15	A checksum error was encountered on an input record from the library file.
16	A checksum error was encountered on an input record from the link ready file.
17	An end-of-file is present on the start of the library file, (i.e., no program is on the library file).
18	An end-of-file is present on the start of the link ready file (i.e., no program is on the link ready file).

3-4 OBJECT TIME TRACE

The Series 6000 Object Time Trace is a versatile debugging tool that allows the user to follow the execution of a program in such a manner as to permit the detection and correction of programming errors. Detection of specified program states is accomplished with a conditional statement in which virtually any combination of states may be examined in any specified order. Trace also provides for the detection of stall alarm and memory restrict violations within the user's program without operational hardware. Foreground debugging is permitted by allowing the user to trace the progress of an interrupt subroutine connected to a specified interrupt level. Active communication between the user and Trace is maintained at all times through the use of the System Operator Communications input key.

Trace is in link module format and must be added to the processor file using the following procedure:

- 1) Place the link module on the link ready file (15) via the \$INCLUDE Job Control Statement.
- 2) Link Trace using the following statement:

\$LINK x,y,nrs

DOS-II General Specification

where:

x = background low
y = background high
nrs = starting address of the non-resident service area.

These three parameters may be obtained by using the program INFO or by noting a map of the operating system. (Non-resident service area is listed as an external name.)

- 3) Add the result of this link to the processor file via the following statements:

```
$EDITPF  
ADD
```

These procedures must be followed each time a new DOS is generated.

All trace input statements are received from the System OCD. Prior to inputting any statement, the Operator Communications key (rub-out) must be depressed. If more than 72 characters are needed to complete an input statement, then a semi-colon (;), followed by a carriage return, should be issued and the remainder of the statement be input on the next line. Any characters input after the semicolon and before the carriage return will be ignored. This process may be continued any number of lines until the statement is complete. If an error is made on a continuation line, then only that line need be re-input. For example, the following five lines of input:

```
ABCD;  
FGH  
EFGH;  
IJK;XXX  
LMN
```

are equivalent to:

```
ABCDEFGHIJKLMN
```

The program to be traced must reside on the link ready file (15) in link module format (i.e., the binary output of an assembly or compilation). All options pertinent to the link loader are to be set and the following job control statement is to be issued:

```
$LINKTR
```

The "link trace" statement causes the command processor of DOS to search the processor file directory for the name "TRACE". If the name is not found, the message "ICS" will be output to the System OCD and a hold condition will ensue. On finding the name "TRACE", the program size (program hi-program low) will be added to the address of the NRS (start of Non-Resident Service area) and the resulting sum will be passed to the link loader as the link-low parameter.

Linking will then take place in the usual manner with all applicable messages and options. On completion of the linking process, the load module of Trace residing on the processor file will be loaded overlaying the NRS area and a portion of the background area. Control will then be passed to Trace which will open the list output logical device (06), perform various initializations, and then wait for an input statement from the System OCD.

If the user desires to test an interrupt subroutine that was not generated into DOS, the following procedure is performed:

- 1) Place the main program and the externally-defined interrupt subroutine on the link ready file via the \$INCLUDE statement.
- 2) Issue the following Job Control statements:

```
$OPTIONS .7
$LINKTR
```

- 3) Trace is now in control and the following directives should be issued:

```
ALL
I x,250YYYYY
RES
```

where: x is the address of the dedicated memory location of the group/level to which the interrupt routine is to be subsequently connected and YYYYYY is the memory address of the entry point of the interrupt routine. (This address may be obtained by referring to the link map output by the link loader.) After this linkage has been established, a "CON" directive may be issued.

For additional information, refer to Object Time Trace, General Specification, AA61544.

3-5 DEBUG

Debug provides the ability to input or output memory locations or registers in an octal format, set and reset program breakpoints, and scan memory for specified data configurations. Debug should be loaded only after the background program has been loaded. Background program loading may be accomplished with either the \$LINK or the \$LOADGO Job Control statement. If a \$LOADGO statement is used, an Abort must be issued via the X-OFF control key since execution takes place after loading. (The Abort does not disturb the memory in which the program is now residing.) Regardless of which statement is used to load the background program, the starting address of the program must be noted. Debug is now to be loaded via a \$DEBUG or \$LOADGO DEBUG Job Control statement. Once loaded, Debug requests an input statement from the keyboard (device 0). The statement is then processed and another input statement is requested unless the previous statement specified otherwise. To begin background program execution, a "c x" command must be issued to the starting address of the program. Only memory locations within background low and background high may be specified as the operand of Debug input, continue, or set break commands.

Debug error messages are as follows:

- "ICS" - illegal control statement
- "RAR" - restricted address reference
- "BSO" - break stack overflow

A list of valid control statements is presented in Table 3-3.

3-6 LINK LOADER

The link loader is a special unrestricted processor of DOS that resides in the non-resident service area when invoked. It processes link format records as produced by the DC-6024 Macro Assembler or the DC-6024 FORTRAN IV Compiler and produces, in memory, a program that will be executable when re-loaded via a \$LOADGO Job Control statement.

3-6.1 \$LINK Statement

When linking is to be performed, four parameters are passed to the Link Loader. These parameters are either specifically enumerated on the \$LINK Job Control statement, preassumed values, or a combination of both. The parameters may be specified in any one of five different forms:

- 1) \$LINK
- 2) \$LINK a
- 3) \$LINK a,b
- 4) \$LINK a,b,c
- 5) \$LINK a,b,c,d

where a,b,c and d are octal memory addresses with the following designations:

- a = link low address
- b = link high address
- c = execution low address (relocation bias)
- d = common base address

Table 3-4 shows the values of the parameters for each form of the \$LINK statement.

Table 3-3. Debug Control Statements

Statement	Definition
SRA address	<p>This statement establishes a relative address bias for subsequent address specification. The base address is added to any address that is terminated with an "R".</p> <p>For example</p> <p style="text-align: center;">SRA 20000 I 135R,0</p> <p>causes the contents of memory location 20135 to be set to zero.</p>
I address,data,data,...	This statement ("Input") causes the system to accept octal data constants and store them in sequential memory locations, starting from the specified address. If the specified address is outside the background limits, an error message is typed and the statement is ignored.
O low,high	This statement ("Output") causes the contents of consecutive memory, from the specified "low" through the specified "high", to be typed in octal, one word per line.
OD low,high	This statement ("Octal Dump") generates an octal dump to the list output device. The format is an address, followed by 8 words of data per line.
OA low,high	This statement ("Output ASCII") types a line of ASCII text from the specified "low" through the specified "high". The text is limited to 24 words (72 characters).
SB address	This statement ("Set Break") sets a breakpoint at the specified address. Eight breaks (maximum) are allowed at one time. A breakpoint is set by saving the specified address and its contents in a dedicated stack and replacing the contents of the address with a "BSL break". When a break address is executed, the break routine saves registers I, J, K, E, A, and C, types "ADDRESS", and waits for a new command.
RB	This statement ("Reset Breaks") causes all breaks and the stack pointer to be reset.
IB	This statement ("Initialize Breaks") initializes the breakpoint stack. This should be executed prior to using "SB" and "RB" to avoid including superfluous breakpoint information in the program.
C address	This statement ("Continue") without an address specification causes registers I, J, K, E, A, and C to be restored from the last break. The instruction that belongs in the break address is then executed (not restored) and the background process is continued at the break address plus one. If an address is specified, registers are restored and control is transferred directly to the specified address.
RC address	This statement ("Reset and Continue") without an address specification causes the last breakpoint to be reset, restores registers I, J, K, E, A, and C, and continues the background process at the last break address. If an address is specified with the statement, control is transferred directly to that address rather than the break address.
OR IJKEAC	This statement ("Output Registers") causes the contents of the specified registers (saved from last break) to be output in octal. If a register specification is not included, all registers are output.
OR XY	<p>This statement causes the contents of the SAU registers X and Y to be output in octal as follows:</p> <p style="text-align: center;">X 12345670 123456 123 Y 123</p> <p>where X is expressed as the most significant mantissa, least significant mantissa, and the exponent; and Y is the 3 digit condition register.</p>
X 12345670,123456,123	This statement sets the most significant mantissa, the least significant mantissa, and the exponent of the SAU-X register, respectively. Register Y is set as a function of X.
Y 123	This statement sets register Y as specified without changing X.
OR VH	<p>This statement causes the contents of the Boolean registers V and H to be output in octal as follows:</p> <p style="text-align: center;">V 123456 H 1</p>
V 123456	This statement sets the 16-bit V register as specified.
H 1 0	This statement sets the 1-bit H register as specified.
I data J data K data E data A data C data	These statements cause the specified octal data to replace the current contents of the respective register (refer to Statements C and RC above).
Z low,high	This statement ("Zero") causes the contents of memory, from the specified "low" through the specified "high", to be set to zero.
S low,high,word,mask	<p>This statement ("Scan") causes the contents of memory, "low" through "high", to be searched for the following conditions:</p> <p style="text-align: center;">(Memory .AND. Mask) .XOR. word 0</p> <p>Each time the condition is true, the memory address and its contents are typed. For example, S 0, 10000, 173, 77777 compares only the least significant 15 bits of each word (0-10000) with 173; effectively, listing all reference to 173, disregarding command bits.</p>
EXIT	This statement causes control to be returned to Job Control via the system linkage EXIT.

Table 3-4. Values of Link Parameters

FORM	Link Low	Link High	Execution Low	Common Base
\$LINK	Background Low	Background High	Background Low	Background High
\$LINK a	a	Background High	a	Background High
\$LINK a,b	a	b	a	b
\$LINK a,b,c	a	b	c	b
\$LINK a,b,c,d	a	b	c	d

The area between link low and link high is used to develop the program. The memory area outside these bounds is in no way modified by the Link Loader. The program as it is being linked is built upward, starting at the link low address. The external table created by the Link Loader is built downwards, starting at the link high address. If a relocation bias is specified (execution low), then the program will be linked between link low and link high relative to execution low. If the specified execution low address is not the same as the link low parameter (this is referred to as a biased load), then the resulting program that resides between link low and link high may not be executed. Rather, the program must be dumped via the \$DUMP Job Control statement or cataloged on the processor file via the \$EDITPF Job Control statement. Once dumped or cataloged, the program may now be reloaded at the execution low address and be executed. The common base parameter is used as the initial common base (refer to Paragraph 3-6.3).

3-6.2 Linking Across the 32K Boundary

Provision is made in the Link Loader for linking programs across the 32K memory partition. If during the linking process a module crosses the 32K boundary, the linking process is re-started for that module (and that module only) at the beginning of the upper memory partition ('100000). This is accomplished with the DOS I/O function code "set current record address" (17) and hence, logical device 15 must be assigned to a disc file (generally '15, the link ready file). If the program being linked contains assembly language modules, then care should be taken that external and common requests are made with 16-bit instructions and address constants. If the program being linked was compiled with the DC 6024 FORTRAN IV Compiler with option bit 9 set, only 16-bit external and common requests are generated; therefore, linking will proceed unimpeded.

When a link is performed in a machine with more than 32K of memory, the link parameters must be such that a biased load (execution low .NE. link low) is not performed across the boundary. In addition, biased loads may not be performed entirely within one of the two memory partitions if the execution low parameter is such that the linked program would overlap the memory boundary at execution time.

3-6.3 Common Memory Allocation

The common base parameter of the \$LINK Job Control statement specifies the memory address to be used as the initial common base. When a common definition is encountered, the common base is reduced by the size of the common block, and this address is used as the base of the common block. As common requests are encountered, the associated displacement is added to the base of the corresponding block name. The resultant sum then becomes the address of the common request, whether or not the load is biased. The common base parameter need not lie between the link low and link high parameters as data is not actually being loaded, only request addresses are being satisfied (refer to Paragraph 3-6.4 for an exception to this). The only requirement is that the common base be such that, when the program is loaded for execution, the program does not overlap the common area. Hence, the area between the link low and link high address need only be large enough to contain the program per se and the link loader's external table. Note also that the common area may be assigned above or below a program as long as no overlap occurs at execution time. A unique advantage is gained in assigning the common base parameter below the program (specifically, at execution low); i.e., the common area will be compressed against the program. However, Common should not be assigned below a FORTRAN program that is to be loaded across the 32K boundary as common variables are accessed via indexed references and must, therefore, be assigned above the referencing program.

The non-resident service area in which the Link Loader resides may also be chosen as the common area. However, this area should not be used for the common area if multiphase programs are to be linked that are ultimately to be loaded via the DOS CHAIN loader. This is due to the fact that the non-resident service area is used to contain the processor file directory.

3-6.4 BLOCK DATA Subprograms

The BLOCK DATA subprogram is used to initialize data in a Common block. Since memory is actually modified within the common area when this module is processed, the common base parameter must be between the link low and link high specification. (The only area modified at link time is the region between link low and link high.) In addition, care must be taken that sufficient memory is available between the common block being modified and the link high specification in order to contain the link loader's external table. (Five locations per external, common, and external equivalence definition are required.)

BLOCK DATA subprograms must always be linked from the link ready file (15) and may never reside on the Library file (12). This is due to the fact that there is no absolute manner of determining the correspondence between the program being linked and a BLOCK DATA subprogram if the BLOCK DATA subprogram is on the Library file.

3-6.5 Options

The DOS Link Loader recognizes two option settings (six and seven) as follows:

Option 6:	reset -	Causes END\$ records to be ignored on logical device 15.
-----------	---------	--

- set -- Causes the message "END\$" to be output to the Operator Communications Device on encountering an END\$ record on the Link Ready file. A release (control key BELL) must then be issued to continue. This option is generally used when logical device 15 is assigned to a paper tape input device.
- Option 7:
- reset -- On successful completion of the linking process, exit will be made to the operating system.
- set -- On successful completion of the linking process, a listing of the Link Loader's external table will be output to the list output logical device (06) before exit is made to the operating system.

3-6.6 Link Map

On successful completion of the linking process, a listing of the Link Loader's external table, which is referred to as a Link Map, will be output to the List Output logical device (06) if option bit seven is set. Two side-by-side columns will be output which are in alphabetic and numeric sequence, respectively. The link parameters that are returned to the operating system and other special information is also output. The number of lines output per list output page is controlled by an operating system parameter. This parameter is set via the \$LINES Job Control statement.

The format of the Link Map is as follows:

P. NAME = L L L L L L

XXXXXX	YYYYYY	ZA	XXXXXX	YYYYYY
XXXXXX	YYYYYY	ZA	XXXXXX	YYYYYY
:	:	:	:	:

where:

LLLLLL is the name of the program (as defined by the first encountered NAME definition).

XXXXXX is the external, external equivalence, or common block name.

Z is a single character identifier which may be one of the following:

- 1) (blank) - indicates that the XXXXXX name is a Link Loader parameter
- 2) C - indicates that the XXXXXX name is a common block.
- 3) \$ - indicates that the XXXXXX name is an external.
- 4) # - indicates that the XXXXXX name is an external equivalence definition.

A is a single character identifier which may be one of the following:

- 1) (blank) - indicates that the XXXXXX name was not multiply defined.
- 2) M - indicates that the XXXXXX name was multiply defined.

The link parameters that are passed back to the operating system also appear as an XXXXXX name with a special format. The first character of the XXXXXX name of a link parameter is a special character ("*" or "."). These names and their meanings are as follows:

- *LOW The YYYYYY value associated with this name is the low memory address of the program when loaded for execution (not necessarily link low). If the common area is below the program, then the address of the lowest common element will be output.
- *HIGH The YYYYYY value associated with this name is the high memory address of the program when loaded for execution. If the program contains references to common that were assigned above the program, then the common area will not be included in the high address. However, if the common area was initialized via a Block Data subprogram then the highest initialized address will be output.
- *START The YYYYYY value associated with this name is the starting address of the program when loaded for execution.
- .BCOMM The YYYYYY value associated with this name is the lowest address of blank common if defined in the program.
- .PASS The YYYYYY value associated with this name is the number of passes through the Library file necessary to satisfy external requests. If this value is greater than one, then the modules on the Library file are not ordered.

All numeric values (YYYYYY) reflect the values which will be used when the program is loaded for execution (refer to the examples in Paragraph 3-6.9 for illustrations of actual Link Maps).

3-6.7 Code Processing

The following paragraphs describes some of the more important codes that are processed by the Link Loader. The processor(s) that produces the code is identified along with usage considerations and the action taken by the Link Loader.

A. External Definition

An external definition is generated by the DC 6024 Macro Assembler when the pseudo-operation "XDEF" is encountered and by the DC 6024 FORTRAN IV Compiler when the statements "SUBROUTINE" or "FUNCTION" are received. This code defines an address that is to

be associated with an external name. The name may be identical to an external equivalence or Common block name without conflict. If external definitions of the same name are encountered in subsequent modules, then linkage will be made to the definition that was encountered first. The module that contains the duplicate external definitions will not be loaded if it is encountered on the Library file unless it also contains another definition that was previously requested but not satisfied. If the module that contains the duplicate external definition resides on the Link Ready file, then it will be loaded regardless. In either case, the fact that an external name was defined more than once will be noted on the Link Map (refer to Paragraph 3-6.6).

A useful application of this functioning is in testing subroutines having the same name as a routine on the Library file by placing the module to be tested on the Link Ready file along with the calling program.

B. External Request

An external request is generated by the DC 6024 Macro Assembler when an operand is encountered that is preceeded by a dollar sign (\$). External requests are also generated by the DC 6024 FORTRAN IV Compiler any time subroutines or functions are determined to be external to the program. External requests are also identified as to whether or not they are to be considered unconditional or conditional requests. Conditional external requests (denoted in assembly language by preceeding the operand by two consecutive dollar signs) are satisfied only if the requested name was previously unconditionally requested. If the name was not unconditionally requested prior to the conditional request, then a BLU instruction to the system service routine \$ABORT will be substituted for the requesting instruction (refer to Paragraph 6-2.1).

C. System Service Requests

A system service request is generated by the DC 6024 Macro Assembler when the instruction BLU \$XXX is encountered. It is also generated by the DC 6024 FORTRAN IV Compiler when certain statements such as "CALL EXIT" are received.

If the requested external name is found in the Link Loader's External Name table indicating that an external definition having that name has already been loaded, then a BLL instruction is inserted and the linkage is satisfied. If the requested external name is not found in the Link Loader's External Name table, and is in the DOS System Service table, a BLU instruction to the associated dedicated memory location is inserted and the linkage is satisfied. If the requested external name is not found in either table, a BLL instruction is loaded and the external name is entered in the Link Loader's External Name table. For the linkage to be satisfied, the requested external name must follow in a module residing on the Link Ready or Library file.

This functioning permits linkage to DOS System Services or user defined routines irrespective of whether or not a particular service is resident within the operating system or on the Library file. In addition, services may be added or deleted from the resident portion of the operating system without re-assembling or compiling the requesting program.

D. External Equivalence Definition

External equivalence definitions are generated by the DC 6024 Macro Assembler when the pseudo-operation "XEQV" is encountered. This definition defines a 24-bit constant which is to be merged with the corresponding external equivalence request when encountered. This functioning is useful in externally defining data constants, channel/unit numbers of input/output instructions, etc. The external equivalence name associated with the definition may be identical to an external or common block name without conflict. If multiple external equivalence definitions of the same name are encountered, the first one will be used as the definition and all subsequent definitions of the same name will be ignored. However, the fact that the name was defined more than once will be noted on the Link Map (refer to Paragraph 3-6.6). External equivalence definitions are similar to Common definitions in that the external equivalence definition must precede any external equivalence requests.

E. External Equivalence Request

External equivalence requests are generated by the DC 6024 Macro Assembler when an operand is encountered that is preceded by a number sign (#). This request indicates that the value associated with the previously encountered external equivalence definition of the same name is to be merged (24-bit 'OR') with the requesting frame.

F. Common Definition

A common definition is generated by the DC 6024 Macro Assembler when the pseudo-operation "COMM" is encountered, or by the DC 6024 FORTRAN IV Compiler when a "COMMON" statement is received. The common definition specifies the size of the common area to the Link Loader so that subsequent common requests may be assigned an address (refer to Paragraph 3-6.3). The definition of the overall size of a labeled common block must be identical in all programs and subprograms in which it is defined. However, blank common areas, defined in the various programs and subprograms to be linked, do not have to correspond in size. The only restriction is that the first blank common definition in a set of programs and subprograms to be linked must be the largest block (this is an American National Standard requirement). The common block name associated with the common definition may be identical to an external name or external equivalence name without conflict.

G. Common Request

A common request is generated by the DC 6024 Macro Assembler or DC 6024 FORTRAN IV Compiler whenever a reference to a variable which has been declared to be in common is made. The request carries the displacement from the common block name with which it is associated. For example, if variables A, B, and C are declared to be in common (labeled or blank), then their displacement from the block name is 0, 1, and 2 respectively. Common requests are distinguished as to whether the address size is 15 or 16 bits. Hence, 15-bit common requests must reference only those variables defined in the same memory bank as the request (refer to Paragraph 3-6.3).

H. Name Definition

A name definition is generated by the DC 6024 Macro Assembler when the pseudo-operation "NAME" is encountered, or by the DC 6024 FORTRAN IV Compiler when a "NAME" statement is received. This name is returned to the operating system and is used as an identification of the program that has been linked. If multiple name definitions are encountered, then the first one encountered will be returned to the operating system and all subsequent name definitions will be ignored.

I. Source Program Error

A source program error code is generated by the DC 6024 Macro Assembler and the DC 6024 FORTRAN IV Compiler when an irrecoverable error is detected. This code will immediately terminate the linking process and an error message will be output.

J. Common Origin

A common origin code is generated by the DC 6024 FORTRAN IV Compiler when a BLOCK DATA subprogram is encountered. This code temporarily resets the Link Loader's relative program location counter so that data may be loaded into a common block (refer to Paragraph 3-6.4 for considerations in using BLOCK DATA subprograms).

K. END Code

An END code is generated by the DC 6024 Macro Assembler and the DC 6024 FORTRAN IV Compiler when an "END" statement is received. This code defines the end of the link module currently being processed and causes the Link Loader to prepare to accept another module.

L. END-Jump Relative Code

An END-jump relative code is generated by the DC 6024 Macro Assembler when an "END" pseudo-operation is received which contains a relative operand expression. This code defines for DOS the relative starting address of the program being linked. If this code is not received, then the starting address is assumed to be the program low parameter which is optionally specified on the \$LINK Job Control statement (refer to Paragraph 3-6.1). If more than one END-jump relative code is received, then the last one encountered will be returned to the operating system.

M. END-Jump Absolute Code

An END-jump absolute code is generated by the DC 6024 Macro Assembler when an "END" pseudo-operation is received which contains an absolute operand expression. This code is identical to the END-jump relative code except that the address is not relocated before being passed to DOS.

N. END\$ Record

An END\$ record is generated by the DC 2024 Macro Assembler or the DC 6024 FORTRAN IV Compiler when an "END\$" statement is received. This record is ignored if option bit 6 is not set (refer to Paragraph 3-6.5).

3-6.8 Error Messages

During the linking process, extensive checking is performed to ensure a proper program load. If an error condition is detected, a message will be output to the List Output logical device (06). The format of the error message is as follows:

LNK XX @ ABSOLUTE YYYYYY, RELATIVE ZZZZZZ IN MODULE AAAAAA **LINK ABORTED**

where:

XX	Is a two-decimal digit error code. (The meanings of the error codes are given in Table 3-5.)
YYYYYY	Is the memory address within the link low to link high bounds that the error occurred.
ZZZZZZ	Is the address relative to the module being linked in which the error occurred.
AAAAAA	Is the last encountered external definition or name definition. If no external or name definitions were encountered, "***MAIN" will be output.

For some of the error messages listed in Table 3-5, the module name AAAAAA could possibly be deceptive and hence, caution should be exercised.

Table 3-5. Link Loader Error Codes

Error Number	Meaning
1	Insufficient room to link the program. (The program is overlapping the external table.)
2	A BLOCK DATA subprogram was encountered with the common base outside the bounds of link low to link high.
3	A 15-bit external request was encountered in the lower 32K memory partition without being satisfied in the lower 32K memory partition.
4	An external definition was encountered twice in the same module.
5	The specified link high parameter is greater than background high.

Table 3-5. Link Loader Error Codes (Cont'd.)

Error Number	Meaning
6	A biased load (execution low is not equal to link low) is being attempted such that the resulting load module would overlap the 32K memory partition.
7	The link parameters are such that a biased load (execution low is not equal to link low) would be performed across the 32K memory partition.
8	The link low parameter is less than background low.
9	An external equivalence request was encountered for an undefined external equivalence name.
10	The specified link low parameter is greater than the link high parameter.
11	An invalid special action code was received. (This generally indicates a bad I/O assignment, the lack of an EOF on the link ready file, or a Segment code.)
12	A link of a program that contains an irrecoverable Source Program error is being attempted.
13	A string boundary violation has occurred. (This generally indicates that an assembly language program has caused the overlay of an external request via an AORG or RORG pseudo-operation.)
14	The size of blank Common has been specified in ascending order.
15	The sizes of identical labeled Common block names do not match.
16	The allocated common region will overlap with the program when loaded for execution.
17	A 15-bit common request was encountered in the lower 32K memory partition for a common variable in the upper 32K memory partition.
18	A 15-bit external request was encountered in the upper 32K memory partition for an external name that was satisfied in the lower 32K memory partition.

Table 3-5. Link Loader Error Codes (Cont'd.)

Error Number	Meaning
19	A word count was not complete when a binary input record was requested. (This generally indicates a bad I/O assignment, a missing EOF on the Link Ready file, etc.)
20	A BLOCK DATA subprogram was encountered on the Library file.
21	A checksum error on a binary input was encountered on either the Link Ready or Library file.

3-6.9 Link Examples

The following examples illustrate some various linking situations. In each, the input link parameters and the list output Link Map is shown.

Example #1

Given the following assembly language program:

```

                                NAME    CAT
                                XEQV    CAT,-1
                                COMM     /CAT/A,B,C
000    CAT    TMA                A
001          AMA                B
002          TAM                C
003          BLL                $CAT
004          BUC                CAT
                                END
                                XDEF     CAT,CAT
                                COMM     /CAT/X,Y,Z
000    CAT    TMA                Z
001          MYA
002          TAM                Z
003          BUC                0,J
004          ZZZ                #CAT
                                END$

```

which when linked with the following Job Control statement:

```
$LINK 20000,20100
```

will result in the following program residing in memory on completion of the link. Note that the program is capable of execution since the link low and execution low parameters are the same.

DOS-II General Specification

' 20000	TMA	' 20075
' 20001	AMA	' 20076
' 20002	TAM	' 20077
' 20003	BLL	' 20005
' 20004	BUC	' 20000
' 20005	TMA	' 20077
' 20006	MYA	
' 20007	TAM	' 20077
' 20010	BUC	0,J
' 20011	DATA	-1

The following Link Map will be output to the List Output logical device if option bit 7 is set:

P. NAME = CAT

*HIGH	20012	.PASS	0
*LOW	20000	*START	20000
*START	20000	*LOW	20000
.PASS	0	CAT	20005 \$
CAT	77777777 #	*HIGH	20012
CAT	20075 C	CAT	20075 C
CAT	20005 \$	CAT	77777777 #

Note that the name "CAT" is used in five different and unique contexts: as a symbolic label, as a common block name, as a program name, as an external name, and as an external equivalence name.

Example #2

Given the following assembly language program:

		NAME	TEST
		COMM	C
000		TMA	A
001		AMA	B
002		TAM	C
003		BLU	\$EXIT
004	A	DATA	0
005	B	DATA	0
		END\$	

which when linked with the following Job Control statement:

\$LINK 20000,21000,1000,21000

will result in the following program residing in memory on completion of the link. Note that it is not capable of being executed in the area of memory in which it is Linked since the execution low parameter is not equal to the link low parameter. However, if the program is cataloged on the Processor file, it will be loaded at location ' 1000 and will then be capable of execution.

' 20000	TMA	' 10004
' 20001	AMA	' 10005
' 20002	TAM	' 20777
' 20003	BLU	2
' 20004	DATA	0
' 20005	DATA	0

The following Link Map will be output to the List Output logical device if option bit 7 is set:

P.NAME= TEST

*HIGH	10006	.PASS	0
*LOW	10000	*START	10000
*START	10000	*LOW	10000
.BCOMM	20777 C	*HIGH	10006
.PASS	0	.BCOMM	20777 C

Note that even though the load is biased that the common addresses are still relative to the common base parameter.

Example #3

Given the following assembly language program:

		NAME	CROSS
000		TMD	ONE
001		TLO	PI
002		BLL	\$F\$DV44
003		BLL	\$ROOTP1
004		BLL	\$LOGM1
005		BLU	\$EXI
006	ONE	DATA	1.0
010	PI	DATA	3.14159
		END	
		XDEF	ROOTP1,A
000	A	TLO	ONE
001		BLL	\$F\$AD44
002		BLL	\$C\$SQRT
003		BJL	0
004	ONE	DATA	1.0
		END	
		XDEF	LOGM1,A
000	A	TLO	ONE
001		BLL	\$F\$SU44
002		BLL	\$C\$LG10
003		BJL	0
004	ONE	DATA	1.0
005		END\$	

DOS-II General Specification

which when linked with the following Job Control statement:

```
$LINK 77762,100100
```

will result in the following program residing in memory on completion of the link.

'77762	TMD	'77770
'77763	TLO	'77772
'77764	BLL	'100214
'77765	BLL	'100000
'77766	BLL	'100006
'77767	BLU	2
'77770	DATA	'20000000
'77771	DATA	'00000001
'77772	DATA	'31103755
'77773	DATA	'00000002
'100000	TLO	'100004
'100001	BLL	'100014
'100002	BLL	'100414
'100003	BJL	0
'100004	DATA	'20000000
'100005	DATA	'00000001
'100006	TLO	'100012
'100007	BLL	'100114
'100010	BLL	'101214
'100011	BJL	0
'100012	DATA	'20000000
'100013	DATA	'00000001

The following Link Map will be output to the List Output logical device if option bit 7 is set:

P.NAME= CROSS

*HIGH	101314	.PASS	1
*LOW	77762	*START	77762
*START	77762	*LOW	77762
.PASS	1	ROOTP1	100000
C\$ERR1	100614	LOGM1	100006
C\$ERR2	100514	F\$AD44	100014
C\$ERR3	100714	F\$SU44	100114
C\$LG10	101214	F\$DV44	100214
C\$SQRT	100414	SQRT	100314
F\$AD44	100014	C\$SQRT	100414
F\$DV44	100214	C\$ERR2	100514
F\$EROR	101014	C\$ERR1	100614
F\$SU44	100114	C\$ERR3	100714
LOGM1	100006	F\$EROR	101014
ROOTP1	100000	C\$LG10	101214
SQRT	100314	*HIGH	101314

Note that there is insufficient room to contain routine "ROOTP1" in the area between the main program and the 32K boundary; hence, routine "ROOTP1" is assigned to '100000.

3-6.10 Input And Code Placement

The following paragraph describes the input format of link module records as produced by the DC 6024 Macro Assembler and the DC 6024 FORTRAN IV Compiler. Also included are the various codes that are accepted by the Link Loader and their placement within the input record.

An input record to the Link Loader is 55 words in length consisting of six 9-word subfields and a one word hash-total checksum as the 55th word of the record. The first word of each 9-word subfield contains eight 3-bit loader codes that determine the action to be taken for each of the following eight words in the subfield (refer to Figure 3-1). Some loader codes require the use of multiple words to describe a particular function, in which case, the codes corresponding to the extra words are set to zero. If word one and word 55 of the input record are set to a minus one and all other words within the record are set to zero, then the record is considered to be an END\$ record.

Table 3-6 lists the various codes which are accepted by the Link Loader and Table 3-7 lists the special action codes.

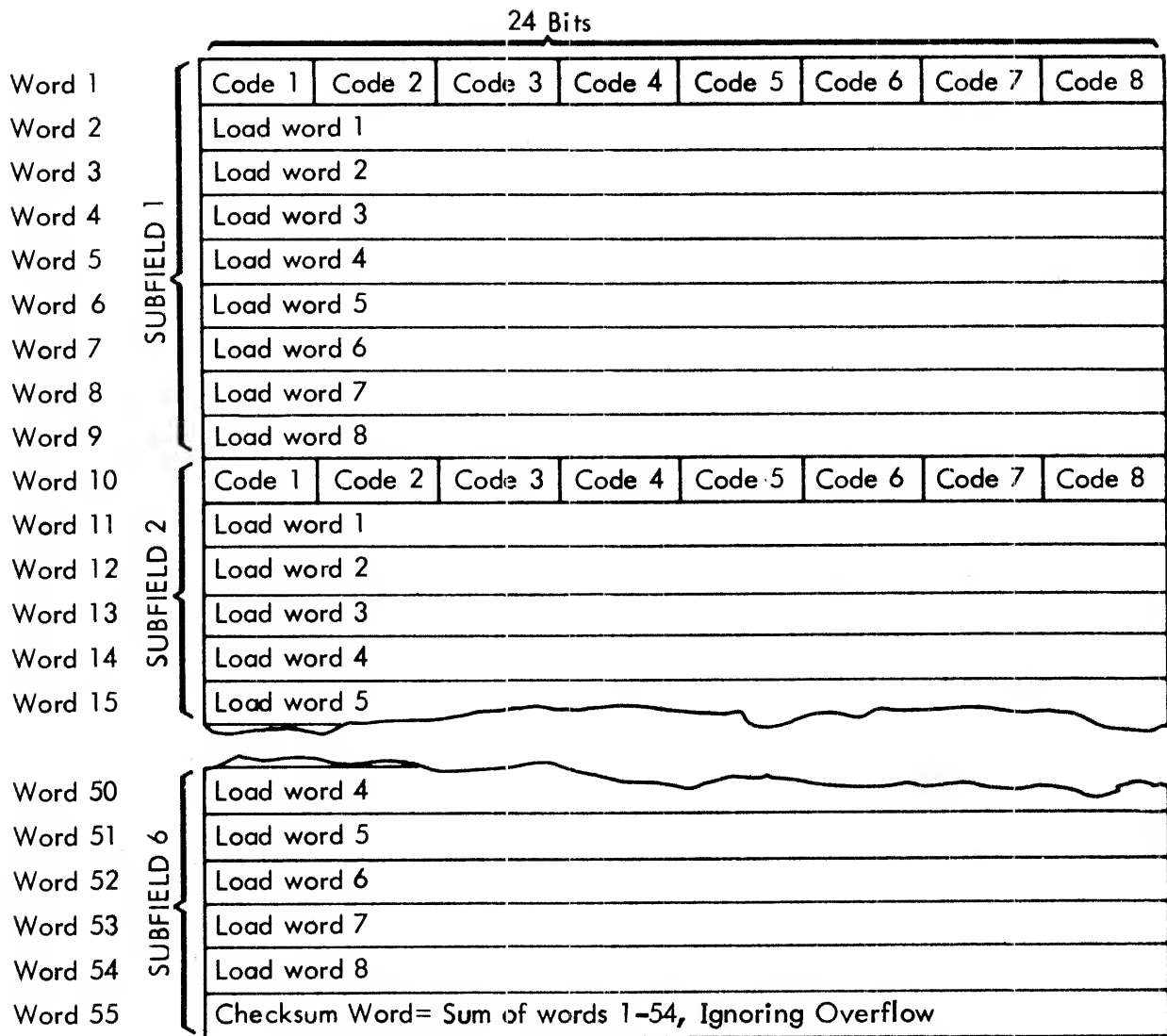


Figure 3-1. Code Placement Format

Table 3-6. Link Loader Input Codes

Code Bit Configuration	Identification and Placement
000	Direct Load.
001	Memory Reference 15-bit.
010	External Definition - the first word contains the address to be associated with the name which follows in the next two words.
011	<p>External Request - the first word is the request frame with the following bit settings:</p> <p style="margin-left: 40px;">B0 = 0: 15-bit request. B0 = 1: 16-bit request. B1 = 0: Unconditional request. B1 = 1: Conditional request.</p> <p>The requested name follows within the next two words.</p>
100	Memory Reference 16-bit.
101	Common Request 15-bit - The address field in the first word contains the displacement from the block name specified in the next two words.
110	Special Action bits 16 through 20 determine the action to be taken (refer to Table 3-7).
111	Common Request 16-bit - The address field in the first word contains the displacement from the block name specified in the next two words.

Table 3-7. Link Loader Special Action Codes

Special Action Bit Configuration	Identification and Placement
00000	ORG absolute - bits 0-15 of the word contain the absolute address.
00001	ORG relative - bits 0-15 of the word contain the relative address.
00010	END
00011	END-Jump Absolute - bits 0-15 contain the absolute address to be passed as the starting address.
00100	END-Jump Relative - bits 0-15 contain the relative address to be passed as the starting address.
00101	Internal String Back - bits 0-15 contain the address of the first link in the chain to be strung.
00110	External String Back - bits 0-15 contain the address of the first link in the chain; the next two words contain the external name.
00111	Name Definition - the next two words contain the name to be associated with the program.
01000	Common Definition - bits 0-15 contain the size of the block; the next two words contain the block name.
01001	Common Origin - bits 0-15 contain the displacement from the block name contained in the next two words into which data is to be loaded.
01010	Source Program Error.
01011	System Service Request - the next two words contain the requested external name.

SECTION IV

BACKGROUND PROCESSORS AND UTILITIES

Background processors and utilities reside on the disc processor file in absolute load module format for quick access. All background programs are I/O independent and request I/O and other system services (e.g., date, lines per page, options, time, etc.) by means of a privileged system service request (refer to Section II).

User written background programs may be added to the processor file through the Processor File Edit routine (refer to Paragraph 3-3). Detailed information pertinent to each standard background processor or utility is provided in the general specification for that product. An example illustrating a typical background job stream follows:

```
$JOB PEANUTS
$DATE XX/YY/ZZ
$ASSIGH 7,7,10,16,6,6,5,5
$OPTIONS .0,7
$ LINES 60
$LOADGO ASSEMBLER
      IDEN  SOURCE PROGRAM EXAMPLE
      NAME SNOOPY
      . . . .
      END$
$WEF 5
$LINK
$EDITPF
ADD SNOOPY
$LOADGO SNOOPY
$EOJ
```

- . End of Program
- . Terminate Binary Link File
- . Link Program into Memory
- . Activate Non-Resident Edit Routine
- . Add Program To Processor File
- . Load and Execute New Program
- . End of Job

SECTION V

LOGICAL FILES AND PHYSICAL DEVICES

5-1 FILE/DEVICE ASSIGNMENTS

A logical file number is a file reference number (00 - n) for input/output processing. The I/O control routine exchanges a given logical file number for an assigned physical device number for executing a table look-up, and then passes the I/O request to the handler that is dedicated to the physical device number.

Physical device handlers are dedicated to a physical device number at system generation time. By assigning (through Job Control) a physical device number to a logical file, the user may switch I/O devices without modifying his program. By assigning the physical device number 0 to a logical file number, the operator may nullify all operations to and/or from that logical file (e.g., suppresses binary output).

5-1.1 File Assignments

A given logical file may have only one physical device number assigned at one time. However, the same physical device number may be assigned to more than one logical file. For consistency between processors and system services, certain logical file numbers are defined and dedicated. These standard logical file numbers (LFN) are shown in Table 5-1.

Table 5-1. Standard Logical File Numbers

LFN (Octal)	Definition
00	Job Stream (JS)
01	Operator Communications (OC)*
02	Undefined
03	Undefined
04	Binary Input (BI)
05	Binary Output (BO)
06	List Output (LO)
07	Symbolic Input (SI)
10	Symbolic Output (SO)
11	Processor File (PF)
12	Link Library File (LF)
13	Link Name File (NF)
14	Compress Source File (SF)
15	Link Ready File (LR)
16 - 22	Disc Work Files
23 - n	Open

*OC may not be reassigned.

DOS-II General Specification

JS (00) is the file number used by Job Control when requesting a job statement. This file may have any input device assigned to it. OC (01) is the file number referenced by Operator Communications (OC), both for input statements and output messages.

BI (04) is the file number referenced by the Include processor in preparing link modules for loading. BO (05) is used by the assembler and compiler to create link load modules. LO (06) is referenced for list output by the assembler and compiler. SI (07) is referenced for symbolic input by all symbolic processors. SO (10) is referenced as a "scratch file" for storing symbolic input records during pass one of an assembly, when option 0 is set. If the "scratch option" is set, pass two input is taken from SO rather than SI. Logical file 10 should be assigned to a disc work file.

Disc storage is divided into 10 logical files (11-22). Files 11 through 14 are restricted system files and 15 through 22 are background work files. File 11, Processor File, has an Absolute Disc Loader (sector 0), a directory (sector 1 through 7), and load modules beginning with DOS (sectors 8 through n), followed by the non-resident services. The background processors on file 11 are also in load module format. These processors may be requested externally through Job Control or internally through the system service CHAIN. The Processor File has a directory that gives direct access to load modules on the file. The standard directory allows 128 entries, each requiring six words. The six words are program name (two words), execution low, execution high, execution start, and disc address.

File 12, Library File, contains subprograms only in link module format. This file is searched by the Link Loader to satisfy external subroutine requests during a linking process. File 13, Name File, is reserved for user programs or subroutines in link module format. File 14, Source File, is for symbolic modules (card image) in a compressed and blocked format. File 15, Link-Ready File, is a temporary work file for preparing link modules for linking or adding to file 12 or 13. This file is rewound by the control statement \$JOB and should be assigned to binary output for compilation or assembly. Files 16 through 22, Work Files, are for general use by background processors.

File extents are established by system generation through the file extents table of the System Linkage Module. Extents may also be temporarily changed via the Job Control statement \$EXTENTS, and will remain changed until the system is reloaded from disc via the bootstrap. A file restrict flag may also be set for a file, permitting data to be protected temporarily. Through system generation, the number of files can be increased, and the extents and restrict flag can be permanently established.

Restricted files are accessible for writing only if the \$ALLOW statement has been given.

5-1.2 Device Assignments

Physical device numbers (PDN) are fixed at system generation according to the order that physical device handler linkages are entered into the Handler Linkage table. With a few exceptions, physical device numbers may be arbitrarily assigned at system generation time. For convenience and consistency between various systems, a suggested Standard Physical Device table is shown in Table 5-2.

Table 5-2. Standard Physical Device Table

PDN (Octal)	Device
01	Console Typewriter*
02	Console Tape Reader*
03	Console Tape Punch*
04	High-Speed Tape Reader
05	High-Speed Tape Punch
06	Line Printer
07	Card Reader
10	Card Punch
11 - 22	Disc Files*
23	Magnetic Tape Transport 0

* Physical device number must not be changed.

5-1.3 Blocked File Handling

Blocking of data files is handled via a resident Blocked File Handler which serves as an interface between a program and a physical device handler. The purpose of the Blocked File Handler is to (1) increase I/O speed by reducing the number of physical read/write accesses and (2) to conserve mass storage by compressing data (i.e., deleting blanks from symbolic data) and blocking data into large records.

Disc storage requirements for symbolic data are reduced approximately 10 to 1 by blocking, and link module data requirements are reduced 2 to 1. By using a block size of 224 words, disc access time is reduced 4 to 1 for link modules and approximately 20 to 1 for symbolic modules.

The handler permits linkage with up to 4 blocked files concurrently. System generation linkage to these files is attained through the Handler Linkage Table at the desired physical device entry. Blocked files must be sequential (e.g., 30,31,32,33) and the symbol BFIPDN must be defined by an external equivalence definition within the System Linkage Module (e.g., XEQV BFIPDN, '30). For example, the following will dedicate blocked files 1-4 to physical device numbers 30-33.

PDH.30	DAC	\$BF1	BLOCKED FILE ONE
PDH.31	DAC	\$BF2	BLOCKED FILE TWO
PDH.32	DAC	\$BF3	BLOCKED FILE THREE
PDH.33	DAC	\$BF4	BLOCKED FILE FOUR

DOS-II General Specification

Because the handler is a buffered interface between the physical I/O and the user, a double ASSIGN is employed. For example, to use blocked file one for symbolic input from disc file 16: \$ASSIGN 7,30,30,16. Blocked handlers may be bypassed by a single assignment (e.g., 7,16); however, a data file that has been generated through the blocked handler should also be accessed through it. The same blocking file need not be used to generate and access a data file. For example, a file may be generated through the assignment 16,30,30,16 and accessed through the assignment 7,31,31,16.

Blocking buffers are sequentially allocated downward from memory high; therefore, the user must reserve area above background via the control statement \$BG. HI n, prior to opening blocked files. (Figure 1 is a memory map where four blocked files are used concurrently.) The standard buffer size is 224 (two disc sectors); however, the user may change this during system specification via the external equivalence table of the System Linkage Module (XEQV BLOKWC, n).

There are two types of data acknowledged by the Blocked File Handler: symbolic and binary. Both types are packed as records within a block. When a block is filled it is written, and data is automatically continued in the next block.

Blocked records are separated by a one-word record gap that contains a 12-bit backward word count and a 12-bit forward word count. A backward word count of '7777 indicates the start of a file (i.e., rewind position), and a forward word count of '0000 is a premature block terminator. If a '0000 forward word count is detected during a read, word one of the next block is assumed to be a new start of record gap.

And End-of-File is a physical EOF record. When a Write-EOF request is received, the current block is terminated and written; then an EOF record is written on the physical device.

Symbolic data compression is accomplished by packing three ASCII characters per word and substituting a single 8-bit "negative blank count" for each string of consecutive "blanks" (ASCII is 7-bit code). For input, symbolic data is decompressed and transferred to the user's buffer at three characters per word until the requested word count is complete.

Suggested file assignments for use of the blocked handler with dedicated system disc files are as follows:

Processor File (Disc File 11)

\$ASSIGN 11,11 . No Blocking is permitted

Link Library File (Disc File 12)

\$ASSIGN 12,30,30,12 . Block File One

Link Ready File (Disc File 15)

\$ASSIGN 15,31,31,15 . Block File Two
\$ASSIGN 5,31 "BO" to Link Ready

Symbolic Output File (Disc File 16)

\$ASSIGN 16,32,32,16	. Block File Three
\$ASSIGN 10,32	"SO" to Symbolic Output

Link Name File (Disc File 13)

\$ASSIGN 13,33,33,13	. Block File Four
----------------------	-------------------

Source File (Disc File 14)

\$ASSIGN 14,14	. EDITSF performs internal blocking and deblocking of file 14.
----------------	---

The blocked handler accepts all the standard functions honored by physical I/O. Unacceptable functions cause the abort message "INVALID BF REQUEST". Pertinent functions are defined as follows:

- | | | |
|-------------|---|---|
| OPEN (07) | - | This function opens the assigned physical device, turns off an internal Write flag, and returns the message "BFOPEN" in register D and the number of sectors per block in K. If sufficient space is not allocated between background high and memory high, the operator message "INSUFFICIENT BLOCK ALLOCATION" is typed. |
| CLOSE (10) | - | If the Write flag is set from the previous request, the current block is terminated and written on the assigned device. The physical device is then closed. |
| REWIND (16) | - | If the Write flag is set, the current block is properly terminated; then the assigned device is rewound and blocking pointers initialized for a start of file condition. |
| SETCFA (21) | - | Set Current File Address saves the current record pointers as a start of file address for subsequent repositioning. |
| RPF (11) | - | Reposition File terminates output if the Write flag is set; then resets the current record pointers to the current file address previously saved by a SETCFA or by the detection of a physical EOF during a read. Functions 11 and 21 are employed by the Assembler (for example) to make multiple passes over modules within a single file. |
| SETCRA (17) | - | Set Current Record Address terminates output if the Write flag is set, then sets current record pointers to the values specified as parameter one of the user's parameter list. CRA is assumed to be one word, specifying a relative block number in bits 0-13 and a relative word-of-block in bits 14-23. This is the format returned in register E by a status request. CRA may not be computed reliably for random access; blocked files are sequentially processed. |

DOS-II General Specification

WEOF (06)	-	Write EOF terminates output if the Write flag is set; then writes an EOF record on the physical device.
ADF (13)	-	Advance File terminates output if the Write flag is set; then advances a file on the physical device.
BSF (12)	-	Backspace File terminates output if the Write flag is set; then backspaces one physical EOF.
ADR (15)	-	Advance Record advances blocking pointers forward one record. If the Write flag is set, the block is written; then a normal advance is executed.
BSR (14)	-	Backspace Record moves blocking pointers backward one record, terminating the current block first if the Write flag is set.
SR (01)	-	Symbolic Read decompresses the next sequential record until depleted; the user's buffer is filled with trailing blanks if necessary.
SW (02)	-	Symbolic Write compresses the specified buffer and blocks data sequentially into the assigned blocking buffer.
BR (03)	-	Binary Read transfers the next sequential binary record to the user's buffer.
BW (04)	-	Binary Write blocks the specified user buffer into the assigned blocking buffer.
STATUS (00)	-	Returns the standard I/O status in register A (A23=Busy, A22=WC incomplete, A21=EOF, A15-0=# of words transferred). Current record address pointers are returned in register E (relative word-of-block in E23-14 and relative block number in E13-0). Register C=f(A). Registers I and J contain the relative extents of the assigned disc file (i.e., I=0 and J=last sector# - first sector #).

Blocked File error messages are defined below. A program abort is initiated upon being released.

"INVALID BF REQUEST"	=	Function codes equal to '05 and greater than '21 are invalid.
"INSUFFICIENT BLOCK ALLOCATION"	=	The space between background high and memory high is insufficient for the blocking files.
"USER WC. .GT. BLOCKING BUFFER!"	=	A specified word count greater than the internal block buffer is assumed to be an error.

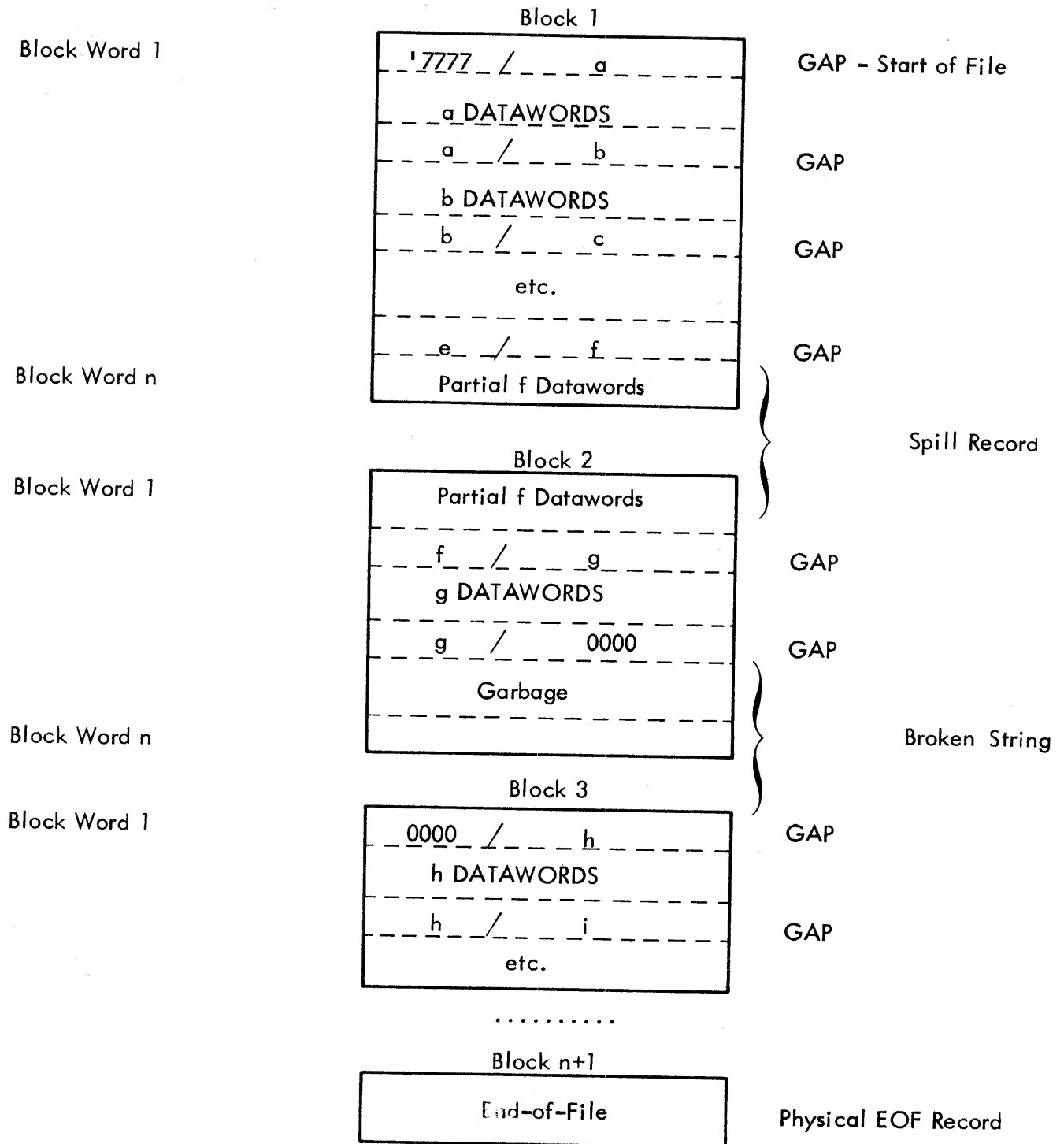


Figure 5-1. Block Structure

DOS-II General Specification

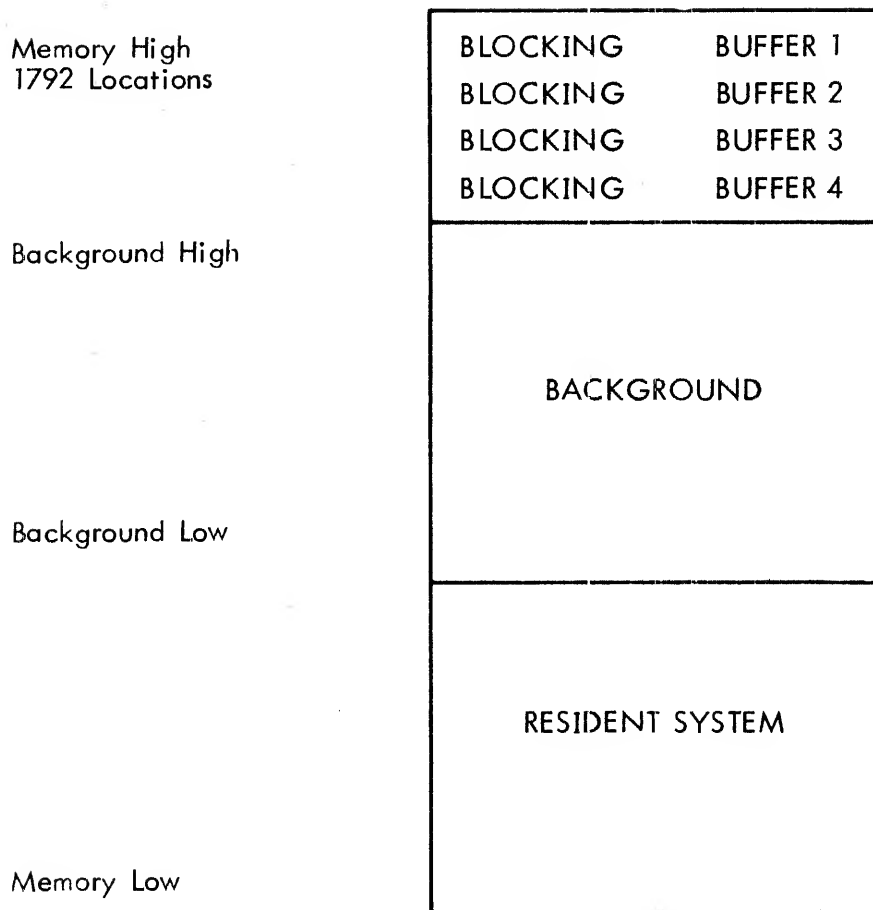


Figure 5-2. Memory Map

5-2 INPUT/OUTPUT FUNCTIONS

The following paragraphs define the valid I/O function codes for each standard device. The 2-digit octal code corresponds to yy in an I/O parameter list. Table 5-3 provides a condensed list of I/O functions and their relative to standard devices.

5-2.1 All Devices

The current status (00) of the referenced I/O handler is returned in register A and the condition register = f (A). If C is negative, status bits A₀ through A₂₂ are not valid. Dedicated status bits are as follows:

A₂₃=ONE = Handler busy (the condition register will be set to "negative").

A₂₂=ONE = Word count not complete.

A₂₁=ONE = End-of-file detected by last request.

A₁₅-A₀ = Number of words actually transferred.

FUNCTION CODE (OCTAL)	DEFINITION OF FUNCTION	ACTION CAUSED BY FUNCTION CODE								
		DISC	CARD READER	LINE PRINTER	PAPER TAPE READERS	PAPER TAPE PUNCHES	MAGNETIC TAPE	CONSOLE PRINTER	CONSOLE KEYBOARD	CARD PUNCH
00	Status	Returns Status of Referenced I/O Handler								
01	Symbolic Read	Read Next Record	Read a Card Convert CARD-CODE to ASCII		Read 8 Channel ASCII and Pack 3 C/W		Read a Record Symbolic Mode Ref: S Tapeop		Accept ASCII Data and Pack 3 C/W	
02	Symbolic Write	Write Next Record		Print a Line		Punch ASCII 3 C/W	Write a Record Symbolic Mode Ref: S Tapeop	Print a Line		Convert ASCII to CARD-CODE and Punch
03	Binary Read	Read Next Record	Read 1 or more cards and pack 2 Col/Word		Read 6 Channel binary and Pack 4 C/W		Read a Record Binary Mode			
04	Binary Write	Write Next Record				Punch 6-Bit Binary 4 C/W	Write a Record Binary Mode			Punch 1 or more Binary Cords (2 Col/Word)
05	Special Action		Read a Card No Code Conversion				Erase 3.5n Inches			
06	Write EOF	Write EOF Record		Top of Form		Punch EOF Record	Write EOF Record	Type EOF Message		Punch 9-8 Code in Column 1
07	Open File	Null	Null	Top of Form	Null	Power ON and Punch 18" Leader	Zero Error Count	Null	Null	Sequence number set to zero
10	Close File	Null	Null	Top of Form	Null	Punch 18" Trailer and Power OFF	Output Error Count	Null	Null	Eject 1 Card
11	Reposition File	Set Next Record Address to Previous EOF + 1					Position tape to End of Previous EOF Record	Null	Null	
12	Backspace File	Set Next Record Address to Previous EOF				Null	Position Tape to Start of Previous EOF Record			
13	Advance File	Scan Past Next Sequential EOF Record	Read Cards Until 9-8 In. Col 1 Detected		Scan Tape for EOF Record		Position Tape to End of Next EOF Record			
14	Backspace Record	Set Next Record Address = Previous Record Address					Position Tape to Start of Previous Record			
15	Advance Record	Scan Past Next Sequential Record	Feed 1 Card		Scan for Next End of Record		Position Tape to End of Next Record			
16	Rewind	Set Record Address to First Record in File	Null	Top of Form	Null	Null	Rewind to Load Point	Null	Null	Null
17	Set Current Record Address	Set Next Record Address From Parameter List					Set Next Record Address From Parameter List			
20	Seek Current Record Address	Initiate Disc Seek For Current Record					Null			
21	Set Current File Address	Set Current File Address = Current Record Address	Null	Null	Null	Null	Set Current File Address = Current Record Address	Null	Null	Null

Table 5-3. Valid I/O Function Codes

1. SHADED BLOCKS REPRESENT ERROR HOLD FOR THE REFERENCED FUNCTION.
2. BLOCKS WITH 'X' REPRESENT OPERATOR MESSAGE AND BACKGROUND HOLD.
3. INTERNAL SYMBOLIC IS 3 ASCII CHARACTERS/WORD.

5-2.2 Disc

The following is a list of the I/O functions and their definitions for the disc:

Status (00) - Register A contains current status. The following registers are also used:

- I = First record address of the specified disc file.
- J = Last record address of the specified disc file.
- E = Current record address of the specified disc file.

The first and last record address are the absolute sector numbers for disc units in the current system. An absolute sector number represents the address of a specific drive, cylinder, track, and sector. The current record and file addresses are relative sector numbers; that is, relative to the start of the file as defined by the first record address.

Symbolic Read (01) - The specified number of words are transferred from the next sequential disc sectors to memory. An end-of-file sector will terminate the transfer; in which case, the number of words transferred is returned in register A, along with an end-of-file status bit, when status is requested (refer to Status (00)).

Symbolic Write (02) - The specified number of words are transferred from memory to the next sequential disc address.

Binary Read (03) - Same as Symbolic Read (01).

Binary Write (04) - Same as Symbolic Write (02).

Special Action (05) - Error Message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Write EOF (06) - An end-of-file sector is written at the next sequential disc address.

Open File (07) - Null.

Close File (10) - Null.

Reposition File (11) - The current record address is set to the address of the record following the previously encountered end-of-file record.

Backspace File (12) - The current record address is set to the address of the previously encountered end-of-file record. Multiple requests for this function will not be affected since only the single previous end-of-file address is saved.

Advance File (13) - A read one word of a sector is repeatedly executed until an end-of-file status from a sector is detected. The current record address and current file address are set to the new address.

Backspace Record (14) - A word count is required for the backspace of a record greater than one sector. The word count of the record is converted to sectors and the current record address is updated accordingly. If the address is outside the file extents, the system will hold with the message "DFO XX". The word count is specified as follows:

```
PARLIST DATA 'XXYY
          DATA WORD COUNT
```

If a short request (TNK 'XXYY) is given, IOC forces the word count to 1.

Advance Record (15) - A word count is required for the advance of a record greater than one sector. The word count of the record is converted to a sector count which determines the number of advances. A read one word of a sector is repeatedly executed until the advance is complete or an end-of-file is detected. The word count is specified as follows:

```
PARLIST DATA 'XXYY
          DATA WORD COUNT
```

If a short request (TNK 'XXYY) is given, IOC forces the word count to 1.

Rewind (16) - The current record address and the current file address are set to zero because record addresses are relative to the given file extents.

Set Current Record Address (17) - Parameter two of the user's parameter list is stored as the current record address of the specified disc file. This function permits random access. The address is relative to the start of the file and is specified as follows:

```
PARLIST DATA 'XXYY
          DATA relative sector address (0-n)
```

Seek Current Record Address (20) - A disc "seek only" is performed for the current record address. Current record address is unchanged.

Set Current File Address (21) - This function causes the current record address of the specified disc file to be stored as the current file address. The primary purpose is to permit the assembler to set the current file address prior to record 1 of an assembly, saving the reposition address for pass two, avoiding the necessity for a scratch file.

DOS-II General Specification

5-2.3 Magnetic Tape

The following is a list of the I/O functions and their definitions for the magnetic tape:

Status (00) - Register A contains current status. The following registers are also used:

K = Current physical end-of-file number of the specified transport.

E = Current record address within the current file.

Symbolic Read (01) - The specified number of words are requested from the appropriate transport. Mode and density are taken from a Tape Options table corresponding to the specified transport (refer to the Control statement \$TAPEOP Table 3-1). For symbolic Read, 3 CPW must always be specified. After completion of the transfer, assuming no errors, the mode bit is tested. If BCD mode was set, a conversion is made within the user's buffer from BCD to ASCII (6-bit BCD for 7-track transport and Extended BCD for 9-track transports). If a read error is detected, ten attempts are made to read the data, then a Hold message is typed (URE T#). If released from the Hold condition, the data is treated as valid.

Symbolic Write (02) - If the BCD mode is specified, the user's buffer is converted from ASCII to BCD (Extended BCD if a 9-track transport is selected) at 3 CPW. The specified number of words are then transferred according to the mode and density specified in the Tape Options table. Ten attempts are made to recover from a write error. The first attempt is in the same spot; subsequent attempts erase three inches first. If this is not successful, a Hold message (UWE T#) is typed; if released, ten more attempts are made.

Binary Read (03) - The specified number of words are read according to the Tape Options table.

Binary Write (04) - The specified number of words are transferred to the selected transport according to the Tape Options table.

Erase (05) - Parameter two of the parameter list is used as a factor where 3.5n inches will be erased.

```
PARLIST DATA 'XXYY
          DATA n
```

Write EOF (06) - End-of-file mark is written.

Open File (07) - Error count is set to zero.

Close File (10) - Error count is typed if not zero.

Reposition File (11) - The tape is repositioned to the first record of the current file (end of the previous end-of-file record).

Backspace File (12) - The tape is positioned to the start of the previous end-of-file record.

Advance File (13) - The tape is positioned to the end of the next end-of-file record.

Backspace Record (14) - The tape is positioned to the start of the previous record.

Advance Record (15) - The tape is positioned to the end of the next record.

Rewind (16) - The tape is repositioned to load point. The current record address and current file address are set to zero.

Set Current Record Address (17) - The tape is repositioned to the record number specified in the parameter list. Load point or an end-of-file is record zero of a file.

```
PARLIST DATA 'XXYY  
          DATA record number
```

Seek Current Record Address (20) - Null

Set Current File Address (21) - This function causes the current record address to be saved as a fake start of file so that a subsequent "reposition file" will reposition the tape to this record address. This permits the assembler, for example, to assemble a string of programs without end-of-file separators, making both passes from tape.

DOS-II General Specification

5-2.4 Console Keyboard

The following is a list of the I/O functions and their definitions for the console keyboard (ASR-33, ASR-35, and KSR-35):

Symbolic Read (01) - Data is accepted on an interrupt basis (1 character per interrupt) until a carriage return (C/R) character is detected or the specified word count is complete. If a C/R is detected before the word count is complete, the user's buffer is filled with blanks. The data is converted to USASCII and packed 3 characters per word as it is received. After the transfer is completed, the first character is tested for Job Control (\$). If a \$ is detected and Job Stream has been previously assigned to the keyboard (JS = 1), the record is passed to the resident command processor for execution as Job Control. After control is returned from the command processor, an input request is reinitialized for the user's buffer, and the background operation is allowed to continue. If an up-arrow (↑) is detected during input, the record is ignored and the input operation is reinitialized. The control key (EOT) is considered to be an End-of-File (EOF).

Symbolic Write (02) - This function generates a Symbolic Write (02) to the console printer.

Binary Read (03) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Binary Write (04) - Same as Binary Read (03).

Special Action (05) - Same as Binary Read (03).

Write EOF (06) - This function generates a Write EOF (06) to the console printer.

Open File (07) - Null.

Close File (10) - Null.

Reposition File (11) - Null.

Backspace File (12) - Same as Binary Read (03).

Advance File (13) - Same as Binary Read (03).

Backspace Record (14) - Same as Binary Read (03).

Advance Record (15) - Same as Binary Read (03).

Rewind (16) - Null.

Set Current Record Address (17) - Same as Binary Read (03).

Seek Current Record Address (20) - Same as Binary Read (03).

Set Current File Address (21) - Null.

5-2.5 Console Printer

The following is a list of the I/O functions and their definitions for the console printer (ASR-33, ASR-35, and KSR-35):

Symbolic Read (01) - This function generates a Symbolic Read (01) request from the console keyboard.

Symbolic Write (02) - The data characters are unpacked and transferred on an interrupt basis. An output record is truncated to 72 characters and all trailing blanks are suppressed. The first character (C_0) in the user's buffer is considered to be a carriage control character. If the first character is an ASCII "1", three leading line feeds are issued as "Top-of-Form". If C_0 is a "0", two line feeds are issued for "double space". If C_0 is a " ", one line feed is used for single spacing and C_0 is discarded. If C_0 is a "+", no line feed is issued for overprinting and C_0 is discarded; otherwise, a single line feed is issued and C_0 is typed to allow operator messages without a carriage control character.

Binary Read (03) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Binary Write (04) - Same as Binary Read (03).

Special Action (05) - Same as Binary Read (03).

DOS-II General Specification

Write EOF (06) - End-of-File message "EOF" typed.

Open File (07) - Null.

Close File (10) - Null.

Reposition File (11) - Null.

Backspace File (12) - Same as Binary Read (03).

Advance File (13) - Same as Binary Read (03).

Backspace Record (14) - Same as Binary Read (03).

Advance Record (15) - Same as Binary Read (03).

Rewind (16) - Null.

Set Current Record Address (17) - Same as Binary Read (03).

Seek Current Record Address (20) - Same as Binary Read (03).

Set Current File Address (21) - Null.

5-2.6 Paper Tape Reader

The following is a list of the I/O functions and their definitions for the paper tape reader (also console tape reader):

Symbolic Read (01) - Same as paragraph 5.2.4, Symbolic Read (01).

Symbolic Write (02) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Binary Read (03) - The binary tape format is four 6-bit frames per word. Each binary record is preceded by an 8-bit line feed (L/F) and terminated by an 8-bit carriage return (C/R). On detection of the L/F, data is accepted and packed in the user's buffer 4 frames per word. On detection of the End-of-Record (C/R), the transfer is terminated.

Binary Write (04) - Same as Symbolic Write (02).

Special Action (05) - Same as Symbolic Write (02).

Write EOF (06) - Same as Symbolic Write (02).

Open File (07) - Null.

Close File (10) - Null.

Reposition File (11) - Message "RPF XX" is typed. ("RPF" is Reposition File and "XX" is the physical device number.) Background is held until the operator performs the function and executes the release command.

Backspace File (12) - Message "BSF XX" is typed. ("BSF" is Backspace File and "XX" is the device number.) Background is held until the operator performs the function and executes the release command.

Advance File (13) - Characters are read and tested for an end-of-file code (EOT).

Backspace Record (14) - Message "BSR XX" is typed. ("BSR" is the Backspace Record and "XX" is the device number.) Background is held until the operator performs the function and executes the release command.

Advance Record (15) - Tape is advanced until the next carriage return (C/R) character is detected.

Rewind (16) - Null.

Set Current Record Address (17) - Same as Symbolic Write (02).

Seek Current Record Address (20) - Same as Symbolic Write (02).

Set Current File Address (21) - Null.

5-2.7 Paper Tape Punch

The following is a list of the I/O functions and their definitions for the paper tape punch (also console tape punch):

Symbolic Read (01) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Symbolic Write (02) - Data is unpacked, converted to ASR-33 code, and transferred on an interrupt basis until the word count is complete. A leading line feed (L/F) and trailing carriage return (C/R) are generated by the handler and transferred.

Binary Read (03) - Same as Symbolic Read (01).

Binary Write (04) - Data is unpacked, four 6-bit frames per word, and transferred to the paper tape. An 8-bit line feed is punched as a start-of-record and a carriage return as end-of-record.

Special Action (05) - Same as Symbolic Read (01).

Write EOF (06) - Punches end-of-file (EOT = '204) code (instead of L/F) as the start code, followed by C/R. Effectively, this generates a zero word record with EOT as start-of-record.

Open File (07) - Punch power is turned on and 18 inches of leader (blank tape) is generated.

Close File (10) - Punches end-of-file (EOT = '204) code, generates 18 inches of trailer (blank tape), and turns off punch power.

Reposition File (11) - Same as Symbolic Read (01).

Backspace File (12) - Null.

Advance File (13) - Same as Symbolic Read (01).

Backspace Record (14) - Same as Symbolic Read (01).

Advance Record (15) - Same as Symbolic Read (01).

Rewind (16) - Null.

Set Current Record Address (17) - Same as Symbolic Read (01).

Seek Current Record Address (20) - Same as Symbolic Read (01).

Set Current File Address (21) - Null.

5-2.8 Card Reader

The following is a list of the I/O functions and their definitions for the card reader:

Symbolic Read (01) - Card input is processed in a manner similar to keyboard input. A 9-8 multipunch in column one is considered to be an end-of-file indicator. A maximum of 80 characters may be transferred by a single request. Hollerith code is converted to ASCII and packed 3 characters per word as it is received. The user has an option of 026 or 029 code conversion.

Symbolic Write (02) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Binary Read (03) - No data is in card columns 1 through 6. Column 1 will be blank except for an end-of-file code (9-8 multipunch) and a partial record code (9-7 multipunch). The partial record or continuation code causes subsequent cards to be fed automatically until one is read without the code. Columns 2 through 6 are open for a sequence number. Columns 7 through 80 are transferred (2 col/word) into the user's buffer until the specified word count is complete or until an end-of-record is reached. If the word count is complete prematurely, cards are ejected until an end-of-record card is reached.

Binary Write (04) - Same as Symbolic Write (02).

DOS-II General Specification

Hollerith Read (05) - 80 columns of data are packed 2 columns/word and transferred to memory without conversion.

Write EOF (06) - Same as Symbolic Write (02).

Open File (07) - Null.

Close File (10) - Null.

Reposition File (11) - Message "RPF XX" is typed. ("RPF" is Reposition File and "XX" is the physical device number.) Background is held until the operator performs the function and executes the release command.

Backspace File (12) - Message "BSF XX" is typed. ("BSF" is Backspace File and "XX" is the device number.) Background is held until the operator performs the function and executes the release command.

Advance File (13) - Cards are fed and column 1 is checked for an end-of-file code (9-8 multipunch).

Backspace Record (14) - Message "BSR XX" is typed. ("BSR" is Backspace Record and "XX" is the device number.) Background is held until the operator performs the function and executes the release command.

Advance Record (15) - One card is fed and ejected.

Rewind (16) - Null.

Set Current Record Address (17) - Same as Symbolic Write (02).

Seek Current Record Address (20) - Same as Symbolic Write (02).

Set Current File Address (21) - Null

5-2.9 Card Punch

The following is a list of the I/O functions and their definitions for the card punch:

Symbolic Read (01) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Symbolic Write (02) - Converts ASCII to card-code and punches.

Binary Read (03) - Same as Symbolic Read (01).

Binary Write (04) - One or more cards may constitute a binary record. Column one of each card is reserved as a special action code. A 9-8 multipunch in column one indicates an End-of-File record (one card). A 9-7 multipunch indicates a partial record. Column 2 through 6 are open for a sequence number.

Special Action (05) - Same as Symbolic Read (01).

Write EOF (06) - A 9-8 multipunch is punched in column 1.

Open File (07) - Null.

Close File (10) - Ejects one card.

Reposition File (11) - Same as Symbolic Read (01).

Backspace File (12) - Same as Symbolic Read (01).

Advance File (13) - Same as Symbolic Read (01).

Backspace Record (14) - Same as Symbolic Read (01).

Advance Record (15) - Same as Symbolic Read (01).

Rewind (16) - Null.

Set Current Record Address (17) - Same as Symbolic Read (01).

Seek Current Record Address (20) - Same as Symbolic Read (01).

Set Current File Address (21) - Null.

5-2.10 Line Printer

The following is a list of the I/O functions and their definitions for the line printer:

Symbolic Read (01) - Error message "E@XXXXXX" is typed, indicating an error at the address defined by "XXXXXX".

Symbolic Write (02) - C₀ is assumed to be a carriage control character and the printer responds in the manner defined in Table 5-4. A "1", "0", " ", or "+" in C₀ is replaced by the appropriate control character as indicated in Table 5-4. The character "↑" is, by hardware definition, a print command and is therefore not printable.

Table 5-4. Line Printer Carriage Control Characters

C ₀	Action
"@" or "+"	0 line advance
"A" or " "	1 line advance
"B" or "0"	2 line advance
"C"	3 line advance
. . .	
"O"	15 line advance
"P" or "1"	Channel 1 advance (Top of Form)
"Q"	Channel 2 advance
"R"	Channel 3 advance
. . .	
"W"	Channel 8 advance

Binary Read (03) - Same as Symbolic Read (01).

Binary Write (04) - Same as Symbolic Read (01).

Special Action (05) - Same as Symbolic Read (01).

Write EOF (06) - Top-of-form.

Open File (07) - Top-of-form.

Close File (10) - Top-of-form.

Reposition File (11) - Same as Symbolic Read (01).

Backspace File (12) - Same as Symbolic Read (01).

Advance File (13) - Same as Symbolic Read (01).

Backspace Record (14) - Same as Symbolic Read (01).

Advance Record (15) - Same as Symbolic Read (01).

Rewind (16) - Top-of-form.

Set Current Record Address (17) - Same as Symbolic Read (01).

Seek Current Record Address (20) - Same as Symbolic Read (01).

Set Current File Address (21) - Null.

SECTION VI OPERATING PROCEDURES

6-1 LOADING PROCEDURES

DOS is loaded from the disc via the hardware loader or the following thumb-in:

<u>Location</u>		<u>Mnemonic</u>	<u>Octal</u>
0		TOA WC	62500012
1		OAW '500	00714500
2		TMA CW	05000011
3		OCW '500	00700500
4		ISW '500	00730500
5		BNZ *-1	22600004
6		QBB B7	00110200
7		BNZ *-3	22600004
10		BUC '20	21000020
11	CW	DATA '40000000	40000000
12	WC	DAC '100	00000100
13	BA	DAC '20	00000020

After the resident DOS is loaded from disc, the Operator Communications Device (OCD), which is generally the console teletype, will print the message "ABORT". Statements will then be read from the OCD. If, for example, the Job Stream device is the card reader, the operator may then assign control to that device as described in Paragraph 6-2.1.

6-2 OPERATOR CONTROL FACILITIES

The operator can manually control the Job Stream of DOS by using two keys of the OCD, X-OFF and BELL.

6-2.1 Abort Procedure

To unconditionally terminate the current job, the operator must actuate the control key X-OFF. This will cause the message "ABORT" to be printed on the OCD and the Job Stream device will then be assigned to the OCD. To transfer back to the main Job Stream device, the following Assign statement must be entered by the operator:

```
$JOB
$ASSIGN 0,7
```

This statement assigns the logical file 00 (Job Stream) to physical device number 7, the card reader (refer to Section V for a description of the logical file numbers and physical device numbers of DOS).

DOS-II General Specification

6-2.2 Hold Recovery Procedure

A hold condition can result from program control or from a Job Control statement. Generally, a message is printed on the OCD when a hold is encountered. The message informs the operator of action required before continuing. To release a hold condition, the operator must actuate the control key BELL.

6-3 USING SYSTEM PROCESSORS

Operating procedures for each processor are documented separately. In general, a processor is called and supported by the following control statements:

```
$JOB PEANUTS  
$DATE XX/YY/ZZ  
$ASSIGN l,p,l,p  
$OPTIONS .i,j,k  
$LINES n  
$LOADGO processor
```

6-4 OPERATOR MESSAGES

Table 6-1 contains a summary and description of the DOS operator messages. For a listing of Link Loader error messages, refer to Table 3-5 of this document.

Table 6-1. DOS Operator Messages

Condition	Message	Action Taken if Released
Abort Background.	ABORT	A release is not necessary. The Job Stream is assigned to the keyboard and Job Control is called from disc.
Job Control encounters an invalid control statement..	ICS	The statement is ignored and another statement accepted.
Invalid load request for processor XXXXXX.	ILR-XXXXXX	Call Job Control from disc.
An erroneous system request has been made at location XXXXXX.	E @XXXXXX	The condition register is set to NEGATIVE and control is returned to XXXXXX +1. This permits an I/O request to be repeated after the parameters are corrected.
Restricted address reference from Debug.	RAR	The statement is ignored.
Rewind device XX is electronically impossible.	REW XX	Control, with a ZERO condition, is returned to the routine making the request.
Backspace a file on device XX is electronically impossible.	BSF XX	Control, with a ZERO condition, is returned to the routine making the request.
Backspace a record on device XX is electronically impossible.	BSR XX	Control, with a ZERO condition, is returned to the routine making the request.
Reposition the current file on device XX is electronically impossible.	RPF XX	Control, with a ZERO condition, is returned to the routine making the request.
Advance a file on device XX is electronically impossible.	ADF XX	The active process is continued without loss of data.
Advance a record on device XX is electronically impossible.	ADR XX	The active process is continued without loss of data.
Attention device 5. Tape is low on punch.	ATN 05	A flag is set and punching is continued.
Attention device 6. Printer paper is low or yoke is open.	ATN 06	The active process is continued without loss of data.

Table 6-1. DOS Operator Messages (Cont'd.)

Condition	Message	Action Taken if Released
Printer parity error detected.	P. E. 06	The print is repeated using "no line advance" for carriage control.
Memory file overflow.	MFO	Abort.
Disc file overflow on physical file XX. A request was made to perform a function outside the bounds of the file.	DFO XX	An EOF status is set. The function is not executed and control, with a ZERO condition, is returned to the routine making the request.
Disc file status bit 2 Moving head-file unsafe. Fixed head-track address error.	DFS 02	Repeat request.
Disc file status bit 3 Moving head-seek error. Fixed head-sector address error.	DFS 03	Repeat request.
Disc file status bit 4 (read error).	DFS 04	Repeat the request.
Disc file status bit 23 (ABC word count not complete on a read or write request with no EOF detected).	DFS 23	Repeat the request.
Refer to Table 3-5.	LNK XX	Refer to Table 3-5.
A command has been given to include a module which is not on the name file.	INC 01	Control is returned to Job Control.
Checksum error by Include.	INC 02	The binary input file is backspaced and an attempt is made to re-read the faulty record.
Refer to Table 3-2.	ELFXX	Refer to Table 3-2.
A command has been given to add a module which is already in the referenced file.	EPF 01 ENF01 ESF 01	The command is ignored and another Edit command is required from the job control file.
A command has been given to Replace or delete a module which is not in the referenced file	EPF 02 ENF02 ESF 02	The command is ignored and another Edit command is required from the job control file.

Table 6-1. DOS Operator Messages (Cont'd.)

Condition	Message	Action Taken if Released
The Name parameter is missing from the command statement when required.	EPF 03 ENF03 ESF 03	The command is ignored and another Edit command is requested from the job control file.
A command has been given to search file 15 for a specific program that is not on the file.	ENF04	The command is ignored and another Edit command is requested from the job control file.
A recoverable read or write error occurred XX times on the tape transport n specified.	XXE Tn	Data is good and operation proceeds normally.
An unrecoverable read error has occurred on the tape transport n specified.	URE Tn	Ignore the error and assume data to be good.
An unrecoverable write error has occurred on the tape transport n specified.	UWE Tn	Ten additional attempts are made to recover.
Physical end-of-tape has been detected.	EOT Tn	Operation proceeds normally.
Transport n is off line.	ATN Tn	Operation proceeds normally.
Disc file restricted for physical file XX. A write request for a restricted file was encountered.	DFR XX	Abort.
Attention! Disc file read only switch is on.	ATN DF	Repeat request.

SECTION VII

SYSTEM GENERATION

7-1 GENERAL

System generation is the procedure used to configure a Disc Operating System to the user's application. A System Generation System (SGS) is provided with each DC 6024. This program is in paper tape bootstrap format. SGS is a resident operating system with I/O handlers for all standard peripheral devices.

7-2 PRELIMINARY PROCEDURES

Prior to generating a system, the user must determine which peripheral devices are to be included in DOS, and the associated channel, unit and interrupt assignment for each device. When the peripheral complement is established, a custom System Linkage Module (SLM) must be created by updating and assembling the standard SLM source program. The modified SLM, after being assembled, becomes the "main" program module of DOS. The SLM contains all the pertinent parameters and external requests for the user's custom system. Information pertaining to the contents and construction of the SLM tables is contained in the following Paragraphs. If the user does not intend to generate a custom system at this time, then the procedure for generating a standard system is described in Paragraph 7-3.

There are ten tables in the SLM. These tables contain various types of information pertaining to the operation of DOS. Eight of the ten tables are core-resident. In an operational DOS, the contents of the Miscellaneous Parameter table, Physical Assignment table, Tape Options table, and Disc Extents table are often modified by Job Control statements, but the remaining tables are not changed by any service or processor. The SLM tables are described in the following paragraphs.

7-2.1 External Definition Table

The External Definition (XDEF) table contains external definitions for parameters within the SLM. This table permits other DOS modules to make use of the SLM parameters; e.g., the I/O handlers and programs of the System Service Module use many of the parameters in the XDEF table. If the user's foreground modules require inter-communication links, the names should be entered in the XDEF table.

7-2.2 External Equivalence Table

The External Equivalence (XEQV) table defines absolute channel/unit number for each peripheral in the system (refer to Appendix B). It also defines the number of words per sector, sectors per track, tracks per cylinder, and cylinders per drive for the disc units in the system (refer to Appendix A). These constants are externally referenced by the I/O handlers and satisfied at system generation time.

7-2.3 Service Linkage Table

The Service Linkage table is contained in memory locations 0-37₈. These locations can be accessed only by a Branch and Link Unrestricted (BLU) instruction by a restricted processor. The contents of the table are assigned at system generation time. The Service Linkage table contains unconditional branch (BUC) instructions to the various service processors listed in the table. Any location in the table not containing a BUC to a processor must contain a BUC to the system error routine. The referenced processor may be internal or external to SLM. Standard service linkage assignments are shown in Table 7-1 (refer to Paragraph 7-2.8, System Service Directory, for programming considerations).

Table 7-1. Standard Assignments, Service Linkage Table

Linkage Address (Octal)	Identification	Function
0	ABORT	Abort current background job.
1	I/O	I/O control routine.
2	EXIT	Return to Job Control.
3	HOLD	Output operator message and wait.
4	CHAIN	Load absolute program module.
5	INFO	Return system information.
6	O/M	Stack an operator message and continue.
7-37	OPEN	To be defined by user.

7-2.4 Miscellaneous Parameter Table

Memory address 40₈ - 57 are arbitrarily used to define miscellaneous system parameters relative to a given system. Table 7-2 shows a list of parameters used by the standard DOS. The user may supply other entries for use in a custom system.

Table 7-2. Miscellaneous Parameter Table

Parameter	Format	Definition
BG. LOW	DAC \$BGAREA	Background Low
BG. HI	DAC '17777	Background High
MEM. HI	DAC '17777	Memory High
DF1PDN	DAC '11	Disc File One is PDN 11 ₈
SPIL	DATA B8B9...Bn	System Priority Interrupt Levels
OCIL	DATA B8	Operator Communica- tions Interrupt Level
DISCIL	DATA B14	Disc Interrupt Level
CHA	BUC \$C026	For 026 card code conversion.
or CHA	BUC \$C029	For 029 card code conversion.
or CHA	BUC \$C02629	For 026 (\$OPTION 23) or 029 card code conversion.
CBA	BUC \$B:A	For BCD: ASCII request.
CAB	BUC \$A:B	For ASCII: BCD request.
CEA	BUC \$E:A	For EBCD: ASCII request.
CAE	BUC \$A:E	For ASCII: EBCD request.
	BUC \$\$ERR	If no conversion is desired.
F. CP	BUC \$F.DBUG	Operator request link- age to foreground debug.
	BUC 0,J	If linkage is not desired.

The location SPIL (System's Priority Interrupt Levels) is referenced by the service module. SPIL must contain a '1' in each bit position that corresponds to an I/O control system priority interrupt level; e. g., if the system uses levels 8, 9, 10, 12, 13 and 14, then SPIL would be defined as:

SPIL DATA B8B9B10B12B13B14

or

SPIL DATA '73400

It should be noted that foreground interrupt levels are not defined in SPIL. The foreground level must be controlled by foreground executive modules.

7-2.5 Priority Interrupt Linkage Table

Memory addresses 60-1778 are dedicated to the eight executive traps and 72 external interrupt levels. A subroutine call (BSL) is entered for each required level of interrupt to establish linkage with the appropriate interrupt processor. Table 7-3 lists the standard interrupt location assignments.

Table 7-3. Standard Assignments, Priority Interrupt Linkage

Linkage Address (Octal)	Group, Level	Function
60	0,0	Power Down
61	0,1	Power Up
62	0,2	Program Restrict
63	0,3	Instruction Trap
64	0,4	Stall Alarm
65	0,5	Interval Timer
66	0,6	SAU Overflow/Underflow
67	0,7	Address Trap
70-77	1,0-7	Not Assigned
100	1,8	Console Typewriter-Input Processor
101	1,9	Console Typewriter-Output Processor
102	1,10	Card Reader-Input Processor
103	1,11	Card Punch-Output Processor
104	1,12	Tape Reader-Input Processor
105	1,13	Tape Punch-Output Processor
106	1,14	Disc-Ready Status
107	1,15	Magnetic Tape-Ready Status
110-117	1,16-23	Not Assigned
120-147	2,0-23	Not Assigned
150-177	3,0-23	Not Assigned

Executive
Traps7-2.6 I/O Handler Linkage Table

The I/O Handler Linkage table serves as a directory and linkage address table for resident I/O handlers. The order in which the handler address is entered in the table determines the physical device number to which the handler is dedicated. (This is not the same as hardware channel/unit numbers which are defined in the external equivalence table.) For example, the seventh entry in the standard SLM I/O Handler Linkage table is

DAC \$\$. CR

which, by definition, assigns the card reader handler to physical device number 7 (refer to Section V for standard physical device number assignments).

An external address reference (i.e., DAC \$NAME)) to each handler required for the system must be contained in the handler linkage table. A zero must be entered in each location where the device associated with the physical device number is not present; e.g., the seventh entry in the table would be

DAC 0

if the system does not have a card reader. Table 7-4 shows the standard assignments contained in the I/O Handler Linkage table. Note that the console typewriter is assigned to physical device number 1 to facilitate operator communications. This assignment must not be changed; furthermore, Job Control does not allow reassignment of logical bit 1.

Table 7-4. Standard Assignments, I/O Handler Linkage Table

PDN	External Request	Description
1	DAC \$\$. CT	Console Typewriter
2	DAC \$\$. CTR	Console Tape Reader
3	DAC \$\$. CTP	Console Tape Punch
4	DAC \$\$. PTR	Paper Tape Reader
5	DAC \$\$. PTP	Paper Tape Punch
6	DAC \$\$. LP	Line Printer
7	DAC \$\$. CR	Card Reader
10	DAC \$\$. CP	Card Punch
11	DAC \$\$. DFH2	Disc File Handler
	
23	DAC \$\$. MT0	Mag Tape Zero

The \$ASSIGN statement (refer to Table 3-1) relates the Logical File Number (LFN) of a device to a physical device number handler. The physical assignment table is modified (by Job Control) to reflect the current Job I/O assignments. For example, when the statement

\$ASSIGN 0,7,4,15

is encountered, the '7' is entered in the Physical Assignment table at location PAT+0 and the '15' at PAT+4 (assigning physical device number 7 to logical file number 0 and physical device number 15 to logical file number 4). When IOCS obtains a logical file number from a calling

DOS-II General Specification

program, it is used as an index pointer to retrieve the assigned physical device number from PAT. The physical device number then becomes an index pointer to the Handler Linage table which contains a linkage address to the associated physical device handler.

7-2.7 Physical Assignment Table

The Physical Assignment table (PAT) contains the physical device number assigned to logical files 00-n in the standard SLM. Logical file 01 must be assigned to physical device number 01 for operator communications. Other assignments may be preset as desired for system generation.

7-2.8 System Service Directory

The System Service Directory contains the 3-character name for each entry in the Service Linkage table (Table 7-1). The entries must be ordered to correspond in both tables. The System Service Directory is scanned by the Link Loader when it detects a system service request. If the entry is contained in the directory, the loader inserts a Branch and Link Unrestricted (BLU) in the calling program to the appropriate service; if not, a BLU is inserted along with the external request (refer to Paragraph 3-6.7C for a description of the loader's action concerning a system service request).

7-2.9 Disc Extents Table

This table contains the first and last sector number for each disc file. The user may, at system generation, allocate disc storage to suit the requirements of his particular system. Disc allocation may also be modified via the \$EXTENTS statement of Job Control (refer to Table 3-1).

7-2.10 Tape Options Table

The Tape Options table contains definitive information relative to each transport in the system. This information includes type of transport (9- or 7-track), transport number, density, mode (BCD - even parity or Binary/ASCII - odd parity), and characters per word (0, 2, 3, 4).

7-3 SYSTEM GENERATION PROCEDURES

A new disc pack should be initialized with the disc diagnostic before generating a DOS. 1 procedures are given in AA61569-00, Disc Diagnostic, General Specification, Appendix A. This procedure is not necessary prior to each system generation.

7-3.1 SGS Loading Procedure

The System Generation System (SGS) is in paper tape bootstrap format. To load SGS, the bootstrap in Figure 7-1 must be entered in memory and executed. (The automatic bootstrap FILL SWITCH is optional.)

7-3.2 SGS Operating Procedure

SGS starts at location 0 and is self-initializing. The message "ABORT" will be typed when SGS is properly loaded. The program is then in a Hold condition awaiting operator action. When the BELL Key is actuated by the operator, control is transferred to the SGS Job Control processor. Job Stream (LFN = 0) is assigned to the console teletype (PDN = 1). During the system generation process, control will alternate between Job Control and the SGS Link Loader. In the event that the operator needs to acquire control to the system during a Link Loader Hold condition, he may do so by actuating the RUB OUT key (instead of the \$ key) followed by the desired control statement.

Start	TOA	' 110	00	62500110
	OCW	CU	01	00700100*
	IDW	CU	02	00720100*
	COB	0	03	00140000
	BOZ	*-2	04	22200002
L1	TNJ	4	05	63200004
L2	LLA	6	06	00420006
	IDW*	CU	07	00724100*
	BNZ	*-1	10	22600007
	BWJ	L2	11	23200006
	CZA		12	00240020
	BOZ	' 20	13	22200020
	TAM	' 20,1	14	15100020
	BWI	L1	15	23100005
*	CU = 100 for High Speed Reader (Assumed above)			
*	CU = 000 for ASR			
**	Set Sense Switch 1 for ASR Input			

Figure 7-1. Paper Tape Bootstrap

7-3.3 Processor File Development

The Processor file (LFN = 11g) starts at cylinder 0, track 0, and record 0 and is the primary disc file of DOS. The Processor file contains the resident DOS and associated non-resident services in load module format. Development of this file is the function of system generation.

Processor File development procedures, assuming the user starts with an ABORT condition (refer to Appendix C for a sample system generation job stream and associated Link Maps), are as follows:

- 1) Release Job Control by actuating the control key BELL.
- 2) Open a job.

\$JOB SYSGEN

DOS-II General Specification

- 3) Permit access to the restricted Processor File.

\$ALLOW

- 4) Assign binary input, binary output, and list output files to appropriate devices.

\$ASSIGN 4,BI,5,11,6,LO

In SGS, physical device number 11₈ is dedicated to the Processor File; i.e., the area of disc beginning at cylinder 0, track 0, and record 0. BI (binary input) is PDN 4 for the high-speed tape reader and PDN 2 for the ASR reader. LO (list output) is PDN 6 for the line printer and PDN 1 for the ASR printer.

- 5) Set the Link Map option.

\$OPTION .7

- 6) Rewind binary output file.

\$XXYY 0516

- 7) Link the Absolute Disc Loader relative to location 20₈ and dump it in Bootstrap format.

\$LINK 10000,n,20

\$DUMPBF

where: n is background high (37777 for 16K)

- 8) Link the Processor File Directory and dump it in Bootstrap format.

\$LINK 10000

\$DUMPBF

- 9) Link DOS modules SLM, FXM, IOCS, and SSM relative to location 0 and dump DOS in Load Module format. User foreground modules may precede or follow IOCS. SSM must be the last module linked.

The HOLD message, LNK 04, will be typed after each module is linked. When the next module is positioned in the binary input device, linking may be continued by actuating the control key BELL.

When linking is completed, a Link Map will be printed. It will be necessary to reference this map before proceeding.

If linking is complete but all external requests are not satisfied, the map will not be printed. An option 6 may be specified using the operator request key RUB OUT instead of the \$ key. A release, with option 6 set, produces a map of both defined and undefined variables. The problem should be corrected and system generation restarted.

\$LINK 10000,n,0

\$DUMP

- 10) Dump Job Control in Load Module format. The Job Control routine occupies the non-resident area of SSM and is in memory after performing the previous link. The map parameters J. C. R and ENDJCR define the first and last address of Job Control.

\$DUMP J. C. R, a, b, c, a+10000

where: a = first address of the Job Control (J. C. R on Link Map)
 b = last address of the Job Control (ENDJCR on Link Map)
 c = a = starting address of Job Control
 10000 is the base area into which DOS was linked

- 11) Link non-resident system service routines (Link Loader, Debug, Trace, and File Edit) relative to the non-resident service address and dump in load module format. The non-resident service address is defined on the Link Map as NRS.

File Edit contains INCLUDE, EDITPF, EDITLF, EDITNF, and EDITSF on one tape.

Execute the following statements for each service routine, where a = NRS from the Link Map:

\$LINK 10000, n, a
 \$DUMP

- 12) Link any desired background processors relative to BGAREA from the Link Map of DOS, then dump them in load module format. If they do not have a name, specify one in the DUMP statement.

This step may be omitted in preference to Paragraph 7-3.6. It is desirable to perform this step if the system being generated does not have a high-speed reader and if configuration being used for generation does.

Perform the following for each Processor:

\$LINK 10000, n, m
 \$DUMP name

where: m is BGAREA and name is a character identification if the map does not give one.

- 13) Terminate the processor file with an end-of-file record.

\$XXYY 0506

- 14) Initialize the Processor File Directory by linking and executing the IPFD routine.

\$LINKGO

DOS-II General Specification

Once IPFD has been executed, the newly developed DOS is self-supporting. Further initialization of the system may be done under the control of DOS rather than SGS.

Load DOS from disc via the automatic bootstrap fill switch or the thumb-in bootstrap shown in Paragraph 6-1. When DOS is loaded, it will initialize itself, type the message "ABORT", load the non-resident Job Control, and assign Job Stream to the teletypewriter keyboard. The control statement \$JOB must be typed before Job Control will accept any other statement.

7-3.4 Library File Development

Transfer the FORTRAN Support Library to the DOS Library File from paper tape as follows:

```
$JOB
$ASSIGN 4,BI,15,12
$ALLOW
$REW 15
$INCLUDE
$RESTRICT
```

In the Assign statement, BI is the physical device number for the binary input device (4 for the high-speed tape reader and 2 for the ASR reader). Logical file number 15 is the link ready file and physical device number 12 is dedicated to that area of disc allocated to the library file. The Allow statement permits writing on a restricted library file.

Sequential binary records are transferred from paper tape to the disc library file and terminated with an end-of-file record on detecting the END\$ record.

If the library file is to be blocked the sequence should be as follows:

```
$JOB
$ASSIGN 4,BI,15,30,30,12
$BG.HI n
$ALLOW
$REW 15
$INCLUDE
$RESTRICT
```

The double assignment 15,30,30,12 assigns logical file 15 to disc file 12, through Blocked File Handler One ("PDN" 30). The background high specification allows blocking buffer between background and memory high.

If the library file is to be blocked the sequence should be as follows:

```
$JOB
$ASSIGN 4,BI,15,30,30,12
$BG.HI n
$ALLOW
$REW 15
$INCLUDE
$RESTRICT
```

The double assignment 15, 30, 30, 12 assigns logical file 15 to disc file 12, through Blocked File Handler One ("PDN" 30). The background high specification allows blocking buffer between background and memory high.

7-3.5 Name and Source File Initialization

Initialize the link name file (LFN 13) and compressed source file (LFN 14) as follows:

```
$ALLOW
$REW 13
$WEF 13
$REW 14
$WEF 14
$RESTRICT
```

7-3.6 Processor File Expansion

Desired background processors may be added to the processor file. The Assembler was used in the example:

```
$ASSIGN 4,BI,15,15
$REW 15
$INCLUDE
$OPTIONS .7
$LINK
$EDITPF
ADD name
```

The Include statement transfers the link module tape of the processor (ASSEMBLER) to the link-ready disc file and terminates with an EOF record. The Link statement loads the processor into memory at background low and saves the link parameters. The Link statement may also have a relocation address. The Add statement dumps the core resident processor in load module format and enters its load parameter in the processor file directory. If the processor has the desired name, a name need not be given in the Add Statement.

7-3.7 File Listing

After development of the system files, a name list for the contents of the processor and library files may be desired. This may be acquired as follows:

DOS-II General Specification

\$EDITPF
LIST
\$EDITLF
LIST

Refer to Paragraph 3-3 for System File Maintenance.

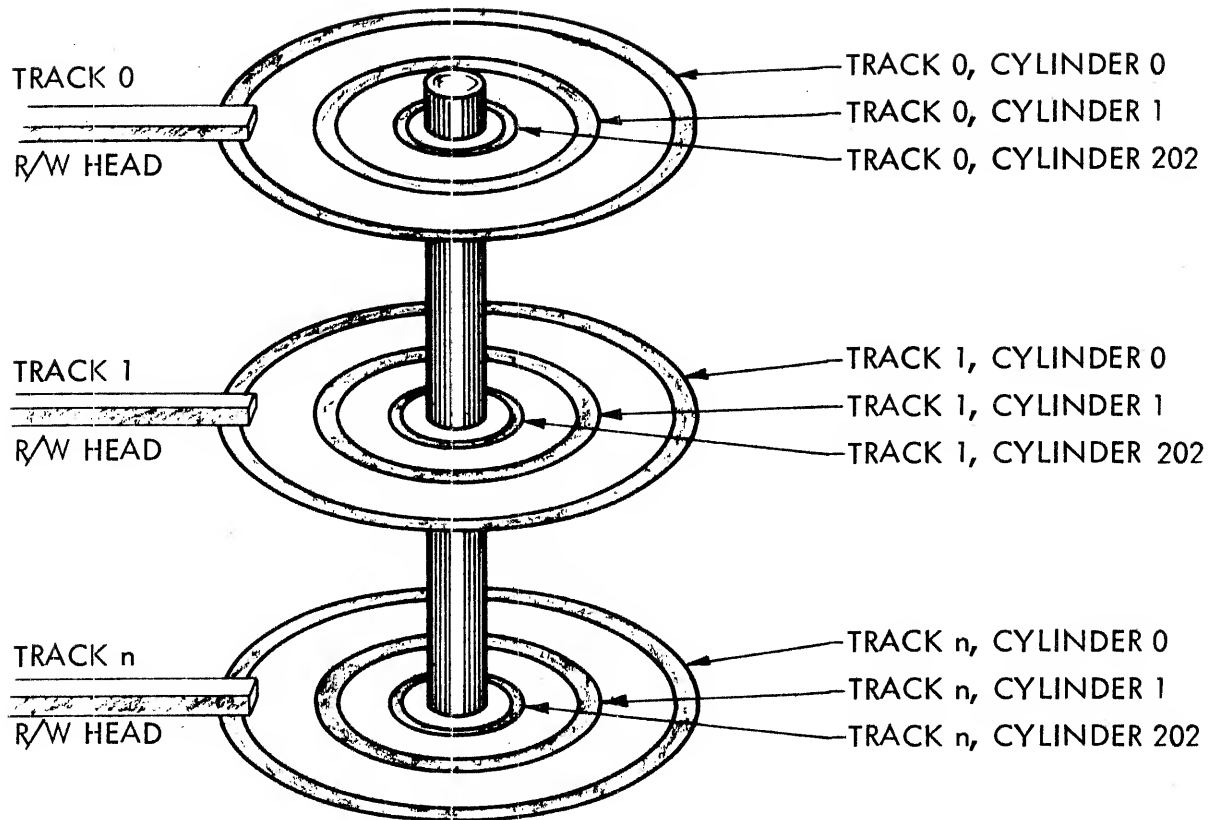
APPENDIX A PHYSICAL CHARACTERISTICS OF THE MOVING-HEAD DISC

The Datacraft Disc Controllers, Models 5100, 5102, and 5104, employ a fixed-size sector of 112 words (refer to Datacraft Manual TM61340-00, Disc Controller) Model 5100 handles 20,300 sectors per drive, Model 5102 handles 81,200 sectors per drive, and Model 5104 handles 162,400 sectors per drive (refer to Table A-1).

DOS uses the absolute sector number for its disc address. An absolute sector number represents the address of a specific drive, cylinder, track, and sector. Data is written on disc as sequential, variable length records (1 to n words per record) in a cylindrical fashion. Data records may spill over sectors within a track, tracks within a cylinder, and from one cylinder into the next adjacent cylinder without program intervention. Figure A-1 shows the disc track/cylinder relationship.

Table A-1. System Characteristics

	5100	5102	5104
Cylinders	203	203	406
Tracks/C	10	20	20
Sectors/T	10	20	20
Words/S	112	112	112
Bytes/W	3	3	3
Bytes/Pack	6,820,800	27,283,200	54,566,400



There are $n+1$ planar surfaces, each of which has a read/write head ($n=9$ for Model 5110 Disc Drive and $n=19$ for Models 5112 and 5114 Disc Drives). The read/write heads are vertically aligned and mechanically locked together. The n read/write heads move incrementally across boundaries called cylinders. There are 203 cylinders per unit for Models 5110 and 5112. Model 5114 has 406 cylinders. A track is a one-plane cylinder and, conversely, a cylinder is comprised of n vertically aligned tracks.

Figure A-1. Cylinder and Track Relationships

APPENDIX B

CHANNEL, UNIT, AND INTERRUPT ASSIGNMENTS

I/O handlers are assigned specific channel/unit numbers and priority interrupt levels. Common assignments are shown in Table B-1 (refer to Paragraph 7-2.2 and Table 7-3 of Section VII of this document).

DC 6024 external interrupts are organized in groups of 24 and are available in subgroups of 4 levels. Priority interrupts for standard peripheral units are assigned to the middle 8 levels of group 1. This provides 8 available levels above, and up to 56 levels below the standard assignment. This permits foreground processors to operate at a lower or higher level than the standard I/O.

Table B-1. Standard Channel/Unit and Interrupt Assignments

Physical Device	Channel/Unit	Interrupt Level
Console Typewriter input/output	0/0	8/9
Card Reader	4/0	10
Card Punch	6/0	11
High-Speed Tape Reader	1/0	12
High-Speed Tape Punch	1/1	13
Line Printer	3/0	none
Disc	5/0	14
Magnetic Tape	7/0	15

APPENDIX C SAMPLE SYSTEM GENERATION JOB STREAM

ABORT
\$JOB SYSGEN
\$ALLOW
\$ASSIGN 4,4,5,11,6,1
\$OPTIONS .7
\$XYYY 0516
\$LINK 10000,17777,20

00020 00062 00020

\$DUMPRF
\$LINK 10000
PFDIR

10000 11100 10000

\$DUMPRF
\$LINK 10000,17777,0
LNK 04
LNK 04
LNK 04
LNK 04

DOS

00000 05253 00000

START 00247	ABORT 00000	EXIT 00002	I/O 00001
SLM 00000	BC.LOW 00040	BC.HI 00041	MEM.HI 00042
S.HLT 00110	S.PAT 00133	S.SSD 00156	S.SSDE 00164
HOLD 00003	INFO 00005	DELTIM 00263	FRA 00164
LRA 00176	CRA 00210	PRA 00222	CFA 00234
DTYPE 00043	S.SPIL 00044	S.OCIL 00045	F.CP 00246
S.IOC 00454	CALLJC 04013	S.HOLD 03541	S.CLDR 03671
S.INFO 03576	S.ERR 00530	BGAREA 05453	POWERD 00265
POWERU 00313	PREST 00374	ITRAP 00414	STALL 00424
TIMER 00356	ATRAP 00404	S.TIP 01120	S.TOP 01422
S.CIP 02422	S.TRIP 01722	S.TPOP 02175	S.DIP 03277
S.CT 00551	S.CTR 00551	S.CTP 00551	S.PTR 01564
S.PTP 02053	S.LP 02771	S.CR 02267	S.DF11 03107
S.DF12 03110	S.DF13 03111	S.DF14 03112	S.DF15 03113
S.DF16 03114	S.DF17 03115	S.DF20 03116	S.DF21 03117
S.DF22 03120	S.BORT 03503	TIME 03653	OPMESS 00774
S.IOE 00530	POW 05204	S.RBG 03564	JCB 04031
S.CP 04003	S.CHA 02607	R.FLAG 04025	SSM 03503
IDLE 03551	J.C.R 04103	ENDJCR 05252	NRS 04103
ABORTF 04024	GOFLAG 04023	USFLAG 04022	OPTION 03645
IRH 03524			

\$DUMP

\$DUMP J.C.R,4103,5252,4103,14103

DOS-II General Specification

SLINK 10000,17777,4103			
LINKLC	04103	05011	04103
SDUMP			
SLINK 10000,17777,4103			
TRACE	04103	10676	04103
SDUMP			
SLINK 10000,17777,4103			
DEBUG	04103	05103	04103
SDUMP			
SLINK 10000,17777,4103			
INCLUD	04103	04364	04103
SDUMP			
SLINK 10000,17777,4103			
EDITPF	04103	05025	04103
SDUMP			
SLINK 10000,17777,4103			
EDITLF	04103	05055	04103
SDUMP			
SLINK 10000,17777,4103			
EDITNF	04103	05054	04103
SDUMP			
SLINK 10000,17777,4103			
EDITSF	04103	11250	04103
SDUMP			
SLINK 10000,17777,4103			
IDISC	04103	05020	04103
SDUMP			
SXXYY 0506			
SLINKGO			
IPFD	07326	07443	07326

ABORT
\$JOB LIBRARY FILE DEVELOPMENT
\$ASSIGN 4,4,15,12
\$ALLOW
\$REW 15
\$INCLUDE
REW 04
\$RESTRICT
\$JOB NAME & SOURCE FILE INITIALIZATION
\$ALLOW
\$REW 13
\$REW 14
\$WEF 13
\$WEF 14
\$RESTRICT
\$JOB PROCESSOR FILE EXPANSION
\$ASSIGN 4,4,15,15
\$REW 15
\$INCLUDE
REW 04
\$OPTIONS .7
\$LINK
\$EDITPF
ADD ASSEMBLER

